НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 122 Комп’ютерні науки та інформаційні технології
на тему Аналіз відеопотоку: ідентифікація аномальних подій

Виконав (-ла): студент (-ка) 4 курсу, групи ТМ-61

Павленко Микола Русланович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник професор кафедри АПЕПС, к.е.н. Сігайов А. О. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент ст. викладач кафедри АЕС і ІТФ, к.т.н Воробйов М.В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

Павленку Миколі Руслановичу
(прізвище, ім’я, по батькові)

1. Тема роботи Аналіз відеопотоку: ідентифікація аномальних подій _____
_____
керівник роботи Сігайов Андрій Олександрович, к.е.н
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № 1168-с

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи Мова програмування Python, СУБД PostgreSQL, бібліотеки PyTorch, Kivy, OpenCV, unittest.
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Проаналізувати сучасні методи вирішення задачі ідентифікації аномальних подій, обрати набір даних і підготувати його, обрати метод вирішення, створити архітектуру нейронної мережі, натренувати модель, оцінити результати, створити структуру бази даних, розробити інтерфейс користувача, провести налагоджування, а також тестування системи.
5. Перелік ілюстративного матеріалу
“Актуальність”, “Постановка задачі”, “Архітектура системи”, “Архітектура C3D”, “Архітектура класифікатора”, “Результати тренування моделі”, “Діаграма прецедентів”, “Концептуальна модель БД”, “Засоби розробки”, “Приклад роботи системи”, “Висновки”.

6. Дата видачі завдання ”11” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	11.10.2019	
2.	Вивчення та аналіз задачі	19.04.2020	
3.	Розробка архітектури та загальної структури системи	26.04.2020	
4.	Розробка структур окремих підсистем	03.05.2020	
5.	Програмна реалізація системи	13.05.2020	
6.	Оформлення пояснювальної записки	07.06.2020	
7.	Захист програмного продукту	8.06.2020	
8.	Передзахист	8.06.2020	
9.	Захист	19.06.2020	

Студент _____ Павленко М. Р.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Сігайов А. О.
(підпис) (прізвище та ініціали,)

Анотація

Дипломна робота: 60 сторінок, 40 рисунків, 3 додатки, 20 джерел.

ГЛИБОКЕ НАВЧАННЯ, КОМП’ЮТЕРНИЙ ЗІР, БІНАРНА КЛАСИФІКАЦІЯ, НЕЙРОННІ МЕРЕЖІ, ЗГОРТКОВІ МЕРЕЖІ, АНОМАЛЬНІ ПОДІЇ.

Об’єкт дослідження — відео, на яких відбувається аномальна або звичайна подія. Предмет дослідження — згорткова нейронна мережа для вилучення ознак з візуальних даних. Мета роботи — розробити систему комп’ютерного зору для автоматичного відеоспостереження із метою класифікації подій на аномальні та звичайні. Актуальність — збільшити рівень безпеки на вулицях, в торговельних центрах, вокзалах, магазинах і т.д. Виконано перевірку навченної мережі, визначено рівень достовірності результатів прогнозування. Шляхи подальшого розвитку предмету дослідження — покращити алгоритм виявлення аномалій, створити централізовану систему із автоматичним звітом у відповідні органи безпеки.

ABSTRACT

Diploma work: 60 pages, 40 figures, 2 appendixes, 20 sources.

DEEP LEARNING, COMPUTER VISION, BINARY CLASSIFICATION, NEURAL NETS, CONVOLUTIONAL NEURAL NETS, ANOMALY EVENT.

The object of research is video data on which happens normal or anomaly event.

The subject of research is convolutional neural network for feature extraction from visual data. The theme: develop computer vision system for automatic video surveillance in order to predict normal or anomaly event. Actuality — increase the level of security on streets, banks, shops, shopping malls, etc. Tested neural network and determined the level of realiability of forecasting results. The further development of system is to increase neural network forecasting and develop centralized system with automatic report to the relevant security authorities.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні скорочення:

MLP — повнозв’язана нейронна мережа.

CNN — конволюційна нейронна мережа.

GPU — графічний процесор.

CPU — процесор.

MSE — середньо квадратична помилка.

MAN — середня абсолютна помилка.

ВСТУП

Камери спостереження все частіше використовуються на вулицях, торгових центрах, банках і т.д для підвищення нашої безпеки. Але моніторинг це задача, яка вимагає колосальних людських ресурсів. Однією із найважливіших задач відеоспостереження є виявлення аномальних подій, таких як дорожньо-транспортна пригода, злочин або незаконна діяльність. Як правило, аномальні події зустрічаються рідко у порівнянні із звичайною людською діяльністю. Саме тому, щоб мінімізувати даремне марнування часу, розробка інтелектуальних алгоритмів відеоспостереження для автоматичного виявлення аномальних подій є насущою проблемою. Метою практичної системи виявлення аномальних подій є своєчасне повідомлення про те, що виникла активність яка відрізняється від повсякденної закономірності, а також індентифікація часового вікна аномалії яка сталася. Невеликим кроком на шляху до виявлення аномалії є алгоритм для виявлення дорожньо-транспортних пригод[1] та кримільних подій. Проте, очевидно такі рішення не можуть бути узагальнені для інших аномальних подій, тому їх застосування на практиці доволі обмежено. Аномальні події в реальному світі є доволі складними та різноманітними. Складно перечислити всі можливі аномальні події. Тому бажано, щоб алгоритм виявлення аномалій не опирався на попередню інформацію про подію. Іншими словами, виявлення аномалій має відбуватися із мінімумом інформації. Існують методи роздільного кодування. Ці методи припускаюсь, що лише мала частина відео — нормальна подія, а решта — аномальна. Однак, оскільки в реальності події змінюються дуже швидко такі підходи призводять до високих показників помилкових спрацьовувань(False Positive).

1. ПОСТАНОВКА ЗАДАЧІ СИСТЕМИ АНАЛІЗУ ВІДЕОПОТОКУ ДЛЯ ІНДЕНТИФІКАЦІЇ АНОМАЛЬНИХ ПОДІЙ

Кінцевою робочою програмою цієї роботи є програмне забезпечення для аналізу відеопотоку із метою розпізнавання аномальних подій на відео в режимі онлайн. Під аналізом мається на увазі класифікація відео на “Звичайні” та “Аномальні”. Оскільки це задача яка важко піддається типовому програмуванню, найкраще за все таку задачу виконувати за допомогою машинного навчання, а саме — глибокого навчання.

Глибоке навчання — це область машинного навчання, яке імітує роботу людського мозку при обробці даних і створює певні шаблони для подальшого прийняття рішення. Зазвичай, ключовим інструментом глибокого навчання є нейронні мережі. Для того щоб створити таку систему необхідно виконанти декілька важливих кроків:

— Збір даних.

— Попередня обробка даних.

— Тренування моделі.

— Тестування моделі.

— Розробка додатку та бази даних.

Данний програмний продукт надаватиме можливість переглядати відео в режимі онлайн з камери відеоспостереження разом із аналізом. Також переглядати базу даних, в якій знаходимуться проаналізовані відео попередніх днів.

1.1 Процес збору даних

Класичний процес збору даних починається із формування вимог до моделі. Іншими словами, що ми хочемо щоб модель класифікувала. У

нашому випадку це “Аномальні” та “Звичайні” відео. Оскільки існує датасет, то мені не довелося робити це вручну.

Датасет — це набір даних у тому чи іншому вигляді. Датасет, який я використовую в даній роботі збирався із платформи YouTube.

1.2 Процес попередньої оброки даних

Цей етап завжди йде після першого етапу збору даних. Суть цього етапу в тому, щоб дані які були зібрані привести до “одного масштабу”. В термінах машинного навчання ми маємо всі дані перевести у числовий вигляд. Відео зазвичай представляють у вигляді картинок, а картинки у вигляді пікселів трьох каналів(RGB). Важливим моментом є те, що потрібно розмітити кожне відео. Розмітка вкрай важлива для задачі з вчителем. По-суті, розмітка це і є вчитель.

Задача з вчителем — це така задача, що для моделі ми даємо набір даних із правильною відповіддю, а вона в свою чергу знаходить такі шаблони в даних, щоб в майбутньому правильно класифікувати нові дані.

Етап попередної обробки даних включає в себе і аугментацію даних, і нормалізацію. Аугментація даних — це збільшення розміру датасету за рахунок деяких маніпуляцій із даними. Наприклад, навмисне зменшення якості відео, поворот на N градусів, обрізання у випадкому місці, затемнення і т.д. Завдяки такій простій операції, отримуємо більший датасет і стійкість моделі до певних зміщень вигляду даних(відео) від очікуваної якості.
Нормалізація даних — це процес приведення даних до одного масштабу для коректної роботи нейронної мережі.

1.3 Процес тренування моделі

Етап тренування моделі є найголовнішим в тому сенсі, що на цьому етапі експерт приймає рішення про тип моделі, її архітектуру, гіперпараметри і т.д. По-суті, задача аналізу відеопотоку — це задача комп’ютерного зору. Існує

безліч методів класичного комп’ютерного зору, але, на жаль, вони не спроможні вирішити добре цю задачу. І це легко пояснити, поведінка людини доволі важко прогнозована річ.

Класичним інструментом для задач комп’ютерного зору в області глибокого навчання є згорткові нейронні мережі. Існують деякі доповнення до цих моделей типу LSTM, Attention і т.д але згорткові мережі — це ключ. Після того, як було вибрано тип моделі, її архітектуру, починається процес навчання. Зазвичай, для тренування моделі потрібен відеопроцесор, оскільки у порівнянні із звичайним процесором, відеопроцесор швидший у 2500 разів, саме для тренування нейронних мереж.

Для порівняння: модель на GPU тренується 3 години, а на CPU тренується два місяці. Звісно, багато чого залежить і від архітектури, і від данних. Але суть залишається одна. Протягом навчання підбираються гіперпараметри(швидкість навчання, кількість епох, розмір батча, періодичність та фактор зменшення швидкості навчання і багато чого іншого). Підбирають спочатку швидкість навчання, а потім все інше. Весь процес навчання контролюється за допомогою графіків та таблиць як для тренувальної, так і валідаційної вибірки.

Валідаційна вибірка — це вибірка, яка не приймає участь у тренуванні, але на ній перевіряється якісь модель під час тренування. Головна мета, з якою роблять цю процедуру — це раннє попередження перенавчання моделі.

1.4 Процес тестування моделі

Тестування натренованої моделі робиться на тестовій вибірці. Тестова вибірка імітує нові, реальні дані але з поправкою, що експерту відома правильна відповідь. Тестова вибірка ніколи не була на процесі тренування, а саме тому ми можемо вважати її репрезентативною. Для задач класифікації це можуть бути різні метрики від простого accuracy до f1-score. Для задач регресії це може бути MAN, MSE чи є ще якась метрика. Вибір цієї метрики

залежить від експерта і від предметної області для якої створюється модель.

Accuracy — доля об’єктів вірно зпрогнозованих відвовідей.

Precision — доля об’єктів, які дійсно відносяться до позитивного класу серед всіх об’єктів, які модель віднесла до позитивного класу.

Recall — доля об’єктів позитивного, які модель зпромоглась знайти відносно всіх об’єтів вибірки.

F1-score — гармонійне середнє між precision і recall

Roc-auc — показує залежність кількості правильно класифікованих позитивних об’єктів від кількості неправильно класифікованих негативних об’єктів.

1.5 Розробка додатку та бази даних

Етап розробки додатку та бази даних передбачає собою створення користувацького інтерфейсу для роботи із програмним продуктом.

Оскільки користувачами цього продукту може бути будь-хто, то головними обосливостями має бути простота та зручність. Було прийнято рішення про створення настільного додатку на базі мови програмування Python та PostgreSQL.

Важливо зазначити, що проектування додатку та бази даних має відбуватися таким чином, щоб у подальшому цей програмний продукт можна було легко підтримувати та розширяти.

2. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Історія розвитку нейронних мереж

В 1943 році нейрофізіолог Уоррен МакКаллок та математик Уолтер Питтс написали статтю про те, як можуть працювати нейронни. Для того, щоб описати як можуть працювати нейрони у мозку вони змоделювали просту нейронну мережу за допомогою електричних схем. В 1958 році Ф. Розенблатт запропонував нейронну мережу, яка зараз називається персептрон та яка створювалася для класифікації об’єктів. При навчанні модель отримувала правильну відповідь від “вчителя”. І завдяки простому правилу ця модель вчилася. Одним із недоліків цієї моделі є те, що для навчання потрібна ідеально лінійно роздільна вибірка. В 1959 році Бернард Уидроу та Марсиан Хофф із Стенфорда розробили модель під назвою “ADALINE”. Перша багатошарова мережа була розроблена в 1975 році. В 90-х роках нейронні мережі створили справжній бум, вони допомагали розв’язувати безліч задач від прогнозування попиту на товар до аналізу платоспроможності клієнтів банку[2]. Останні роки нейронні мережі розвивалася дуже стрімко. Від створення першої моделі AlexNet для класифікації об’єктів на картинках до повністю автономної машини, яка керується нейронною мережею пройшло лише 5 років. Різні технологічні гіганти типу Google, Amazon, Facebook та Apple займаються розробкою власних алгоритмів, фрейворків та систем. Найбільшого успіху в цьому досягла компанія Google — їх модель Bert це справжній прорив у науці.

2.2 Поняття “штучна нейронна мережа”

Під штучною нейронною мережою розуміється математична модель, а також її програмна та апаратна реалізація, побудована по принципу

біологічних нейронних мереж — нервових клітин живого організму. Це поняття виникло при спробі змоделювати процеси, які відбуваються у мозку людини. Штучна нейронна мережа представляє собою систему простих процесів(штучних нейронів), з’єднаних та взаємодіючих між собою. Кожен із процесів мережі має діло з сигналам, які періодично надходять або передаються іншими процесами[3]. Велика мережа спроможна вирішити найскладніші задачі у найкоротші терміни. З математичної точки зору нейронні мережі представляють собою спосіб вирішення нелінійних задач оптимізації. Програмування нейронних мереж передбачає саме навчання мережі, а не написання програмного коду. Завдяки цьому навченна мережа здатна виявляти патерни у даних(вхідними та вихідними), узагальнювати, спрощувати результати та використовувати знання для розбиття складних задач на більш прості.

2.3 Біологічний нейрон та штучний нейрон

Мозок людини та його нервова система складається із нейронів, які з’єднані нервовими волокнами. Між нейронами передаються електричні імпульси за допомогою нервових волокон. Всі дії, які відбуваються із живим організмом, подразнення очей, шкіри, процес мислення — це і є взаємодія між нейронами. Вигляд біологічний нейрон представлено на Рисунку 2.1.

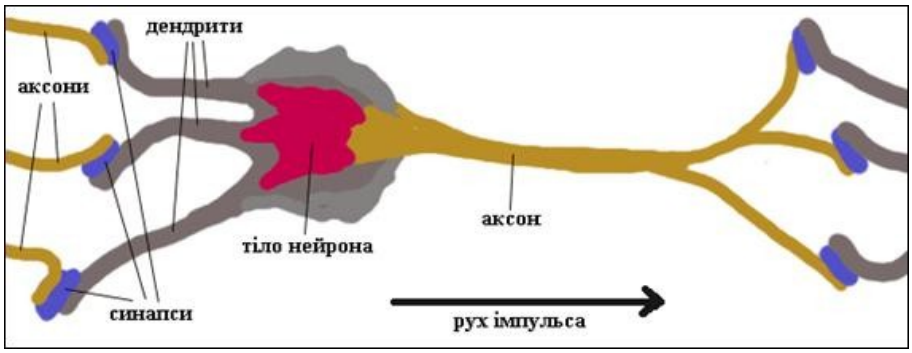

Рисунок 2.1 — біологічний нейрон

— Дендрити — приймають імпульси нейрона

— Аксон — передає імпульс нейрона

— Синапс — утворення, що впливає на силу імпульсу.

Сила імпульсу змінюєтья у деяке визначене число разів, яке ще називають вагою синапса. В тому випадку, коли до нейрона по декільком дендритам надходять імпульси, то вони сумуються. Якщо у сумарного імпульса буде перевищений поріг, то кажуть, що нейрон переходить у стан збудження, формує власний імпульс та надсилає його далі по аксону. Така поведінка нейрона може змінюватися із часом, оскільки ваги синапсів мають властивість змінюватися[4]. Математична модель описаного процесу зображена на рисунку 2.2.

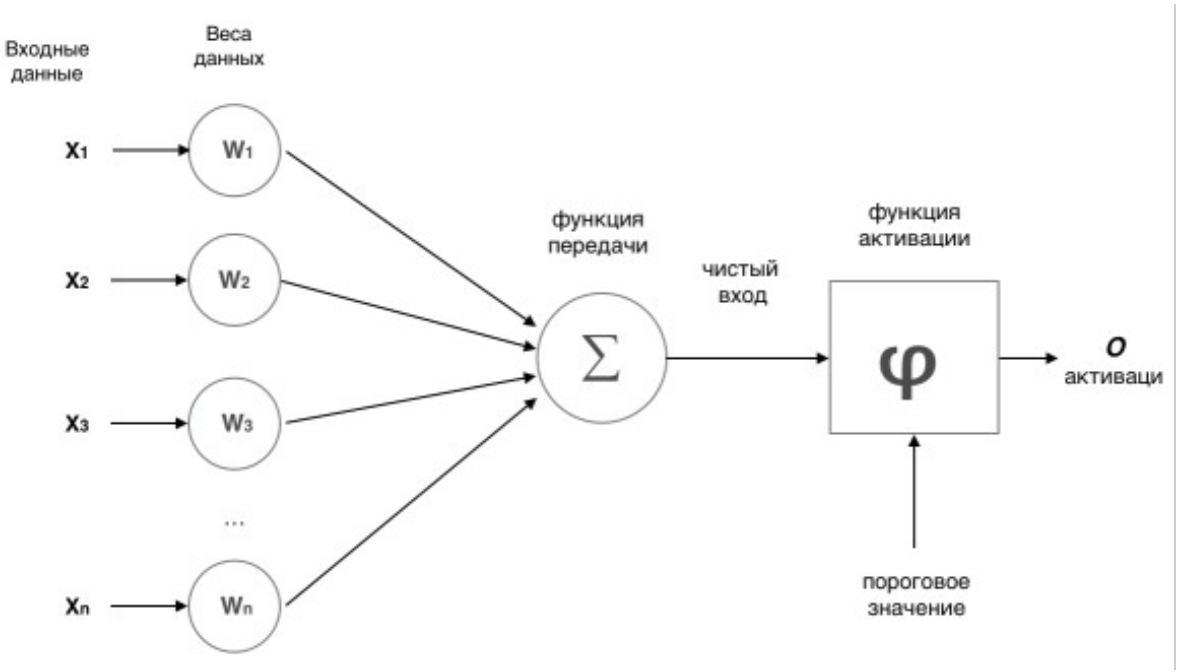

Рисунок 2.2 — архітектура нейрона.

На даному малюнку представлений нейрон із N входами(дендритами), де синапси мають ваги w_1, w_2, w_3, w_n до яких надходять сигнали x_1, x_2, x_3, x_n відповідно. До нейрона надходять імпульси x_1w_1, x_2w_2, x_3w_3, x_nw_n після проходження синапсів та дендритів. Отриманий сумарний імпульс $A = x_1w_1 + x_2w_2 + x_3w_3 + x_nw_n$. За цю операцію відповідає функція передачі. Силою вихідного імпульсу є значення функції активації від значення функції

передачі. Отже сумуючи все вище сказане роботу нейрона можна охактеризувати наступним чином: на його вхід надходить багато різних сигналів(вектор x), кожен із сигналів множиться на відповідну вагу w_i, а результат сумується. Після цього, значення суми передається як аргумент до фунції активації.

2.4 Повнозв’язана нейронна мережа та глибоке навчання

Нейронна мережа або MLP — набір нейронів пов’язані на ациклічному графі. Іншими словами, виходи одних нейронів можуть стати вхідними даними для інших нейронів. У такій системі не може бути циклів, оскільки це автоматично б означало нескінченний цикл при прямому проході по мережі.

Зазвичай такі нейронні мережі об’єднані у окремі шари. Для звичайних нейронних мереж найбільш розповсюдженими типом шару є повнозв’язаний шар, в якому нейрони зв’язані по типу кожен із кожним. Але нейрони всереді одного шару не мають спільних зв’язків. Обов’язоко після кожного шару має бути функція активації. Розмірність вхідного шару має відповідати розмірності ознакового опису об’єкта. Розмірність вихідного шару має відповідати потужності множини відповідей. Якщо деякий об’єкт описати 5 ознакам і якщо Y = {-1, 1}, то $X \in R^5$ $Y \in R^2$. На рисунку 2.3 представлено типову архітектуру такої мережі.

На практиці роблять глибокі нейронні мережі, термін «глибокі» означає два і більше прихованих шарів.

Очевидні переваги глибоких мереж:

— Здатність до апроксимації більш складних функцій.

— Кращий результат на метриках.

— Зменшення часу на пошук шаблонів в даних.

— Модель навченна один раз, може використовуватися до інших задач(з деякими змінами).

А також недоліки:

— Більше часу для навчання.

— Потреба в більшій кількості даних.

— Більша здатність до перенавчання.

— Затухання градієнту при великій кількості шарів.

Незважаючи на всі недоліки, глибоке навчання — провідний інструмент для вирішення багатьох сучасних проблем комп'ютерного зору, тексту і т.д.

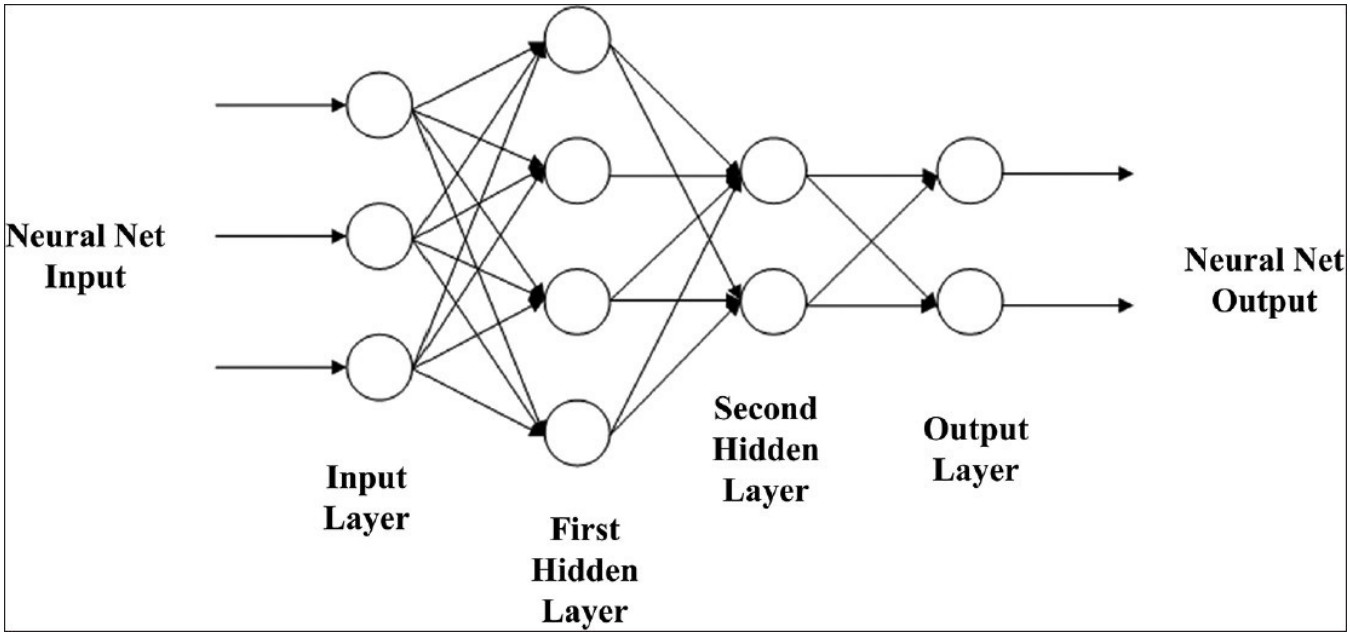

Рисунок 2.3 — повнозв’язана нейронна мережа

В даному випадку розмірність вектора X є R^3. Розмірність Y є R^2.

2.5 Функції активації

Функція активації — це математичний вираз, який визначає вихід нейронної мережі. Аргументом такої функції є значення функції передачі, яка в свою чергу виконує скалярне множення над вектором вхідних даних та матрицею вагів. Однією із головних умов функції активації — це те, що така функція має бути обчислювально ефективною, оскільки обрахунок виконується по тисячам або навіть мільйонам нейронів. Також така функція обов’язково має мати похідну(хочаб першу).

Сучасним методом навчання нейронної мережі є — метод зворотнього поширення помилки. При роботі цього методу йде велике навантаження на функцію помилки та її похідну, тому як вже сказано вкрай важлива обчислювальна ефективність. Головна мета функії активації — надання нелінійності даним. Саме завдяки таким, нелінійнійним функціям нейронна мережа спроможна вчити складні патерни в даних. Неможливо використовувати просту, лінійну функцію активації, оскільки виникає відразу декілька проблем:

- Неможливо використовувати зворотнє поширення помилки для тренування такої моделі оскільки похідна від лінійної фунції активації — константа.
- Втрачає сенс поняття “глибоке навчання” оскільки без нелінійності всі шари можна представити у вигляді одного матричного множення(Композиція афінних перетворень еквівалентна одному афінному перетворенню).

Отже, нижче наведено найбільш розповсюдженні фунції активації та їхня коротка характеристика.

1. Сигмоїд/Логістична функція активації

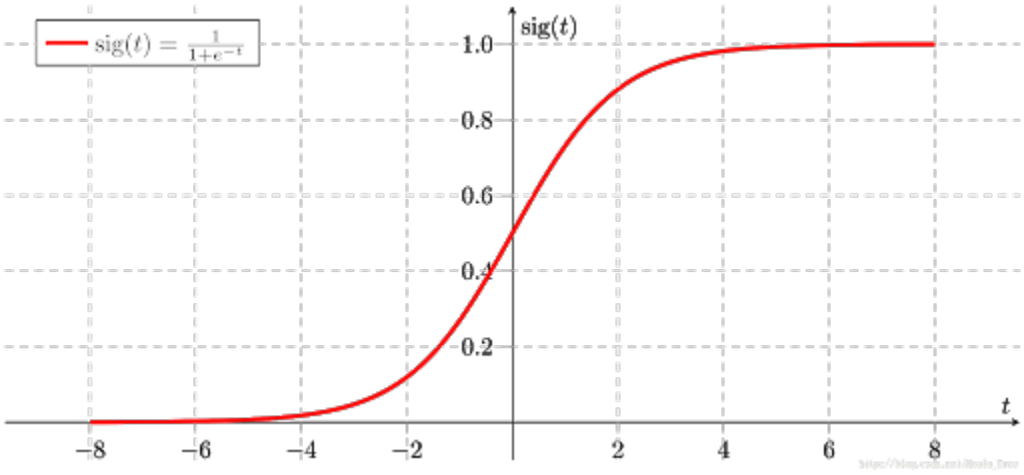

Рисунок 2.4 — функція активації Sigmoid

Переваги:

- Плавний градієнт, який запобігає “стрибкам” у вихідних значеннях
- Границі вихідного значення обмежені від 0 до 1, що дозволяє нормалізувати дані

Недоліки:

- Для великих значень аргументу вихідне значення майже не змінюється, що призводить до затухання градієнту
- Обчислювально важка операція
- Вихідне значення не з нульовим центром

2. ReLU (Rectified Linear Unit)

Переваги:

- Обчислювально ефективна — дозволяє моделі навчатися дуже швидко

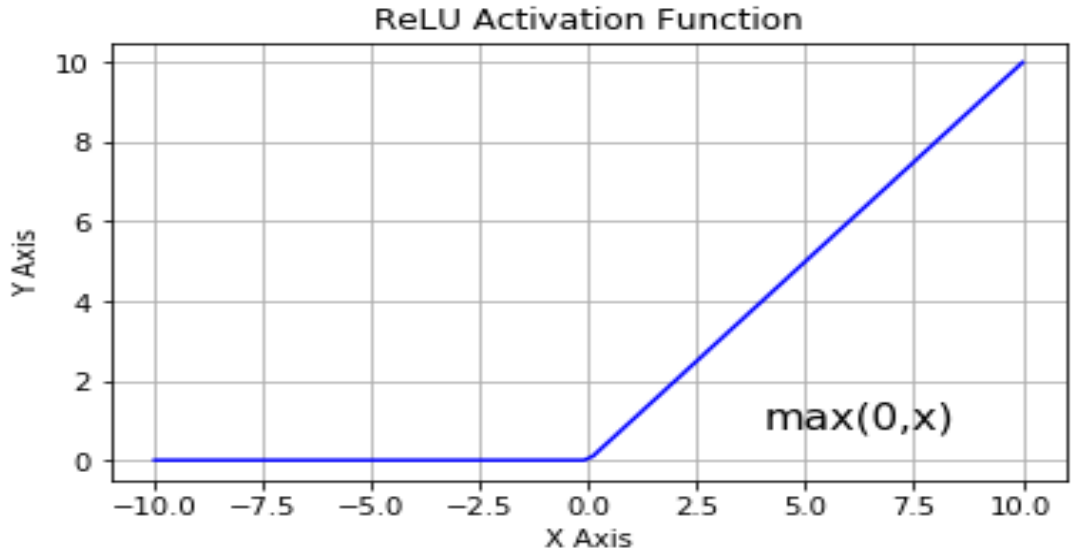

Рисунок 2.5 — функція активації ReLU

Недоліки:

- Якщо аргумент < 0, то градієнт дорівнює нулю. Нейронна мережа не спроможна зробити зворотнє поширення помилки. Така проблема називається “помираюча ReLU”

3. Softmax

Переваги має аналогічно до Sigmod, а також:

- Дана функція призначена для мультикласової класифікації.

Недоліки: аналогічно до Sigmod

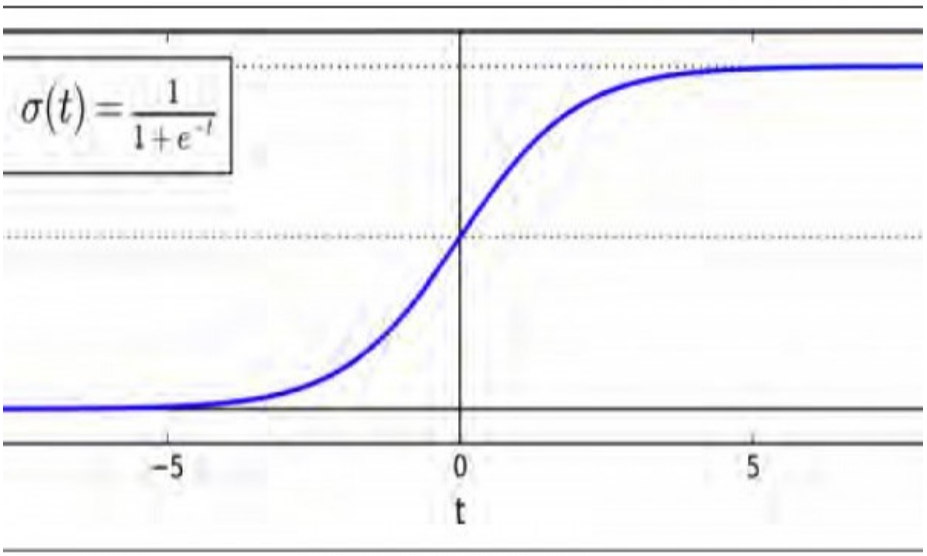

Рисунок 2.6 — функція активації Softmax

4. Hyperbolic Tangent

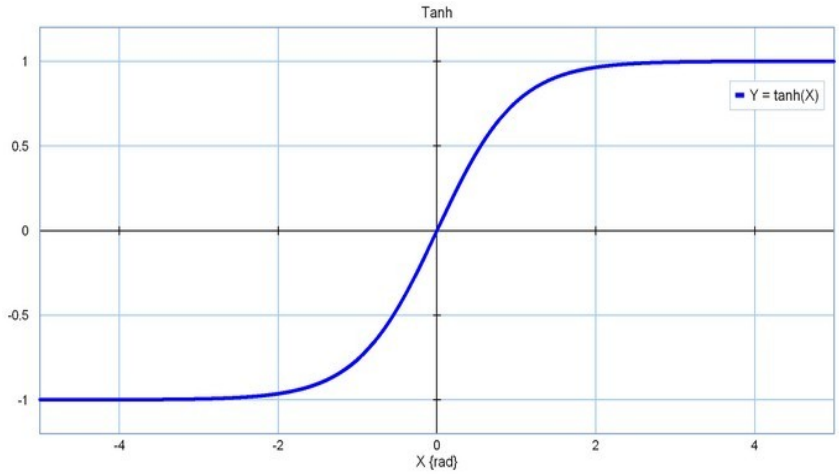

Рисунок 2.7 — функція активації Tanh

Переваги:

- Центр у нулі.
- Інші переваги такі як у Sigmoid.

Недоліки:

- Всі недоліки Sigmod

2.6 Метод зворотнього поширення помилки

Сучасним методом навчання нейронної мережі є алгоритм зворотнього поширення помилки. Типовий процес навчання моделі складається із двох етапів. Перший етап — прямий прохід по мережі. В ході цього етапу обчислюється вихідне значення моделі. Другий етап — обчислення градієнтів та оптимізація функції помилки за допомогою методу градієнтного спуску. Саме другий етап отримав назву зворотній, адже сигнал поширюється від виходу до входу. Більш детально роботу методу розглянемо на прикладі(Рисунок 2.8).

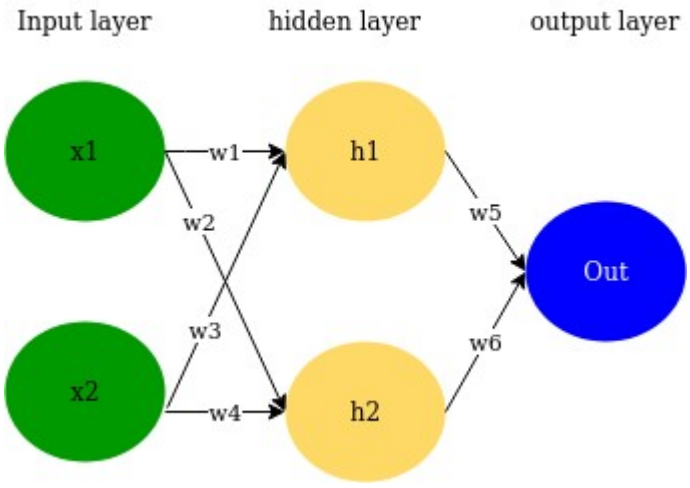

Рисунок 2.8 — проста нейронна мережа

Для спрощення обчислень в даному прикладі не використовується функція активації.

Також нехай ваги мають наступні значення: $w_1 = 0.15$, $w_2=0.24$, $w_3=0.32$, $w_4=0.21$, $w_5=0.18$, $w_6=0.23$. Нехай вхідні дані мають значення: $x_1 = 1$, $x_2 = 5$, а вихідне значення out = 3.

Виконаємо перший етап проходу по мережі(див.формулу 1).

$$\begin{pmatrix}1 & 5\end{pmatrix}\times\begin{pmatrix}0.15 & 0.24\\ 0.32 & 0.21\end{pmatrix}=\begin{pmatrix}1.75 & 1.29\end{pmatrix}\times\begin{pmatrix}0.18\\ 0.23\end{pmatrix}=\begin{pmatrix}0.6117\end{pmatrix} \quad (1)$$

Наступним кроком потрібно порахувати значення помилки.

Для простоти використаємо функцію помилки, яка визначається по наступній формулі(див.формулу 2).

$$Error = 1/2 * (out - actual)^2 \quad (2)$$

Отже, в нашому випадку(диф.формулу 3).

$$Error = 1/2 * (0.6117 - 3)^2 = 3.384 \quad (3)$$

Для того щоб зменшити значення помилки, потрібно змінити вихідне значення мережі. В свою чергу вихідне значення змінюється завдяки вагам, тому ваги напряму впливають на вихідне значення, а значить і на помилку.

Зазвичай, оптимізація виконується за допомогою градієнтного спуску. Цей метод вимагає взяття похідної(градієнту) по параметрам(див.формули 4,5).

$$dError/dW6 = (out - actual) * (x1w3 + x2w4) = -3.27 \quad (4)$$

$$dError/dW5 = (out - actual) * (x1w1 + x2w2) = -3.22 \quad (5)$$

Інші параметри можна знайти аналогічним способом.

Маючи значення похідної для кожного параметра легко оптимізувати функцію за правилом градієнтного спуску(диф.формулу 6)

$$(Wprevious = Wprevious - lambda * dError/dWcurrent) \quad (6)$$

де lambda — це гіперпараметр, який означає як швидкість навчання, а Wcurrent — поточний параметр, який оптимізується[5].

2.7 Згорткова нейронна мережа

Згорткова нейронна мережа також відома як convolutional neural networks або CNN. Даний тип нейронних мереж спеціалізується на даних, які мають “сіточну структуру”. Наприклад, дані пов’язані зі зміною в часі — так звані time-series data, текст(CNN має місце бути, але краще використовувати LSTM архітектури) і звісно картинки. Для таких даних як картинки, відео які в свою чергу можна представити у послідовності картинок, CNN є найкращим вибором завдяки своїй структурі. Назва “згорткова” натякає на те, що

мережа використовує в собі математичну операцію, яка називається “згортка”(англ. convolution). Згортка — це один із видів лінійних операцій.

Згорткова нейронна мережа — це просто мережа, яка використовує згортку замість простого множення матриць хоча б в одному із своїх шарів. Розглянемо більш детально типову архітектуру CNN мережі (Рисунок 2.9).

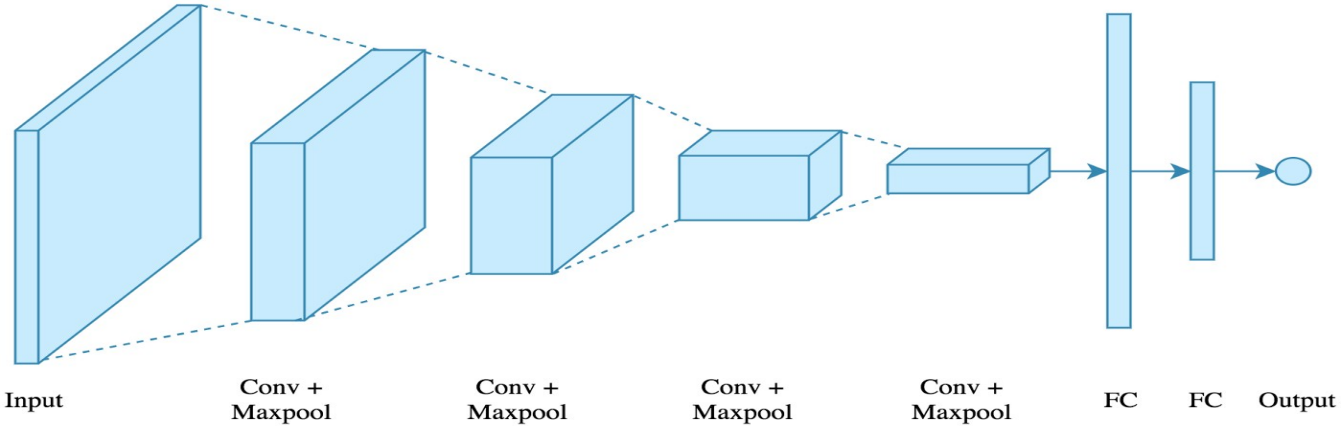

Рисунок 2.9 — типова структура згорткової мережі

• Вхідний шар:

Перший вхід у будь-якого типу нейронної мережі — вхідній. Даний шар зберігає необроблені значення пікселів вхідного зображення. У випадку картинок це деяка кількість значень по ширині, довжині і глибині.

• Шар згортки

Шар згортки — це основний шар в CNN. Даний шар виконує операцію згортки. У випадку картинок згортка проходить наступним чином:
Нехай Input — матриця пікселів для картинки. Kernel — це матриця вагів.

Операція згортки представляє собою поелементне множення пікселя на вагу відповідного скаляру із матриці вагів. Після цього сумується результат. Розмірність результуючої feature map матриці залежить від кількох важливих гіперпараметрів:

• кроку(stride)

• розмірності матриці вагів.

В даному випадку Input це вхідна матриця (Рисунок 2.10), а Kernel

виконує роль вагів.

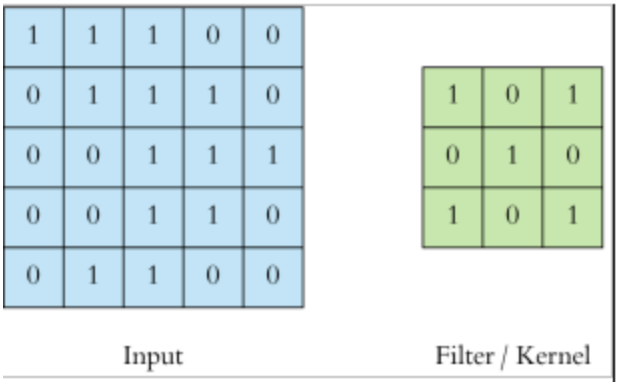

Рисунок 2.10 — приклад матриці вагів та вхідної матриці

Результатом роботи такої згортки є інша матриця ознак (Рисунок 2.11).

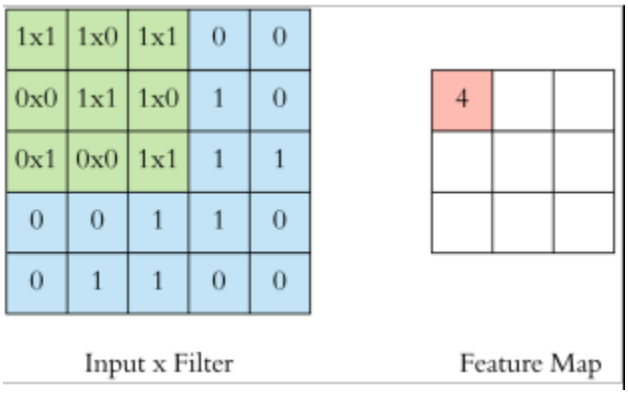

Рисунок 2.11 — результуюча матриця після операції згортки

Завдяки такому алгоритму роботи, на вхід згортки може подаватися майже будь-яка розмірність картинки. Це ще одна із сильних сторін згорткових нейронних мереж.

Очевидні переваги CNN над fully-connected NNS в задачах комп’ютерного зору:

— MLP більш схильні до перенавчання на даних.

— MLP має прокляття розмірності. Наприклад, нехай маємо картинку RGB картинку:

- 32*32*3 = 3, 072 weights

- 200*200*3 = 120, 000 weights
- 1000*1000*3 = 3, 000, 000 weights

— MLP ігнорують опорну точку: не враховують просторову структуру даних, розглядаючи вхідні пікселі, які знаходяться на великій відстані один від одного так само, як пікселі, які знаходяться близько один до одного.

Завдяки свої архітектурі набір шарів згортки дозволяє нам вивчити складні патерни на фото/відео[6].

Фільтри на перших шарах дозволяють вивчити прості патерни:

• лінії

• кут

• кольорові пятна

• точки

Наступні шари дозволяють вивчати комбінацію із попередніх патернів.

На кінцевих шарах згорткової мережі мережа буде мати векторне представлення картинки.

Зазвичай після того як отримують векторне представлення картинки використовують fully-connected NN для конкретної задачі(класифікація, регресія, сегментація і т.д).

Векторне представлення — це величезний прорив, адже завдяки такій простій ідеї можна використовувати модель навченну на величезній базі картинок(наприклад ImageNet) і використовувати її для fine-tuning.

Ідея fine-tuning доволі проста: отримавши векторне представлення картинки ми можемо замінити останній шар моделі на власний, розмірності яка дорівнює потужності множині y* і “довчити” лише останній шар.

• Шар активації

На даному шарі вхідні значення “проходять” через функцію активації. Зазвичай використовуємо Leaky ReLU — вона дозволяє вчити моделі за доволі розумний час.

• Шар субдискретизації або pooling layer

На даному етапі відбувається пулінг отриманих даних, тобто отримуємо

деякі підвибірку. На цій стадії матриця характеристик(feature map) ущільнюється до матриці меншого розміру. На прикладі картинок пуліг зменшує розмірність по ширині та висоті. Таке перетворення зазвичай робиться на унікальних та областях, які не перетинаються. Зазвичай із матриці 2x2 отримуємо один піксель, значення якого максимальне. Також можна робити усереднення та інші операції. Завдяки такій операції модель позбувається від незначних деталей і також частково вирішуємо задачу перенавчання.

• Повнозв’язаний шар

Даний шар представляє собой звичайну нейронну мережу MLP. На цьому етапі обраховується ймовірність належності до класу. Вхідні дані собою представляють собою ніщо інше як векторне представлення картинки [1].

2.8 Аналіз існуючих схожих систем

Безсумнівно, в тій чи іншій формі схожі системи існують. Користуючись відкритими джерелами мені вдалося знайти декілька схожих систем:

— AI Security Cameras – Hikvision

Ця система призначена для розпізнавання облич злочинців в людних місцях, людей зникших без вісті. Плюс такої системи, що можна знайти людей які є злочинцями, але така система не може попередити.

— Agent Vi

Дана система найкраще підходить для міст де існує безперервний потік людей. Плюсом такої системи є те, що вона навчається в режимі реального часу, що дозволяє уникнути ручного налаштування та тестування. Також по словам розробників ця система точно визначає аномалії серед великого скупчення людей.

— Anomaly detection in video using previdctive convolutional long short-term memory networks

Це більше розробка, ніж повноцінна система, але характерною особливістю

такого підходу є використання нейронних мереж із довго-коротко сроковою пам’яттю. Завдяки такому підходу система пам’ятає, що відбулося N часу тому назад і це допомагає збільшити точність розпізнавання. В роботі описано, що автори розпізнають класи типу “група людей”, “направлення проти потоку”, “їзда на мотоциклах” і т.д

Очевидно, у цих систем свої задачі і вони не схожі на дану роботу. Адже більшість класів цієї роботи пов’язані саме з кримінальною активністю.

2.9 Висновки до розділу

В цьому розділі було коротко описано поняття нейронної мережі, її роботу. Також було розглянуто основні типи шарів нейронної мережі для задачі комп’ютерного зору. Встановлено, що у порівнянні із повноз’язаною нейронною мережею згорткова нейронна мережа краще підходить для задачі комп’ютерного зору, а саме — аналіз відеопотоку: індетифікація аномальних сцен. Розглянуто принцип навчання нейронної мережі. Було описано найбільш популярні функції активації для роботи мережі. Маючи розуміння предметної області, легко перейти до читання наступних розділів.

3. ЗАСОБИ РОЗРОБКИ

3.1 Мова програмування Python

Python — об’єктно-орієнтована мова програмування. Модель класів підтримує як поліморфізм, перегрузку операцій, множинне успадкування та інші речі, які притаманні лише ООП. В той же самий час, Python також підтримує і процедурний режим програмування, що робить його ще більш привабливим. Останнім часом мова також почала підтримувати і функціональну парадигму програмування. З точки зору можливостей Python являє собою деякий гібрид. Вбудований інструментальний набір поміщає його між традиційними мовами для написання сценаріїв (наприклад Perl, Scheme) та мовами програмування для розробки систем (наприклад C, C++, Java). Нижче представлено основний інструментальний набір Python[7].

Динамічна типізація

- Python відслідковує види об’єктів, які програма використовує під час своєї роботи. Не потрібно оголошувати типи об’єктів — мова сама про це потурбується. Звідси випливає і мінус такого підходу. Якщо система стає великою, то важко відслідковувати та підтримувати код. Починаючи з Python 3.6 з’явився чудовий інструмент — typing. Ця бібліотека дозволяє явно вказувати розробникам, який тип даних очікує та повертає чи то функція, чи то метод. Завдяки ньому досвідченні розробники на Python не зустрічаються із проблемою описаною вище. Це так званий “синтаксичний цукор”.

Вбудовані інструменти для роботи з даними

- Для обробки всіх типів об’єктів в Python передбачено потужні стандартні операції типу конкатенації колекцій, відображення, зрізи, сортування і т.д

Автоматичне управління пам’яттю

- Цей механізм забезпечує автоматичне виділення пам’яті під об’єкти, а також
- збір сміття.

Бібліотечні інструменти

- Для більш специфічних задач в Python є великий набір готових інстументів, які підтримують майже все, починаючи від регулярних виразів до роботи із мережею.

За допомогою цієї мови програмування можна робити майже все.

Нижче представлені основні, але не всі сфери, в яких використовується Python.

Системне програмування.

- Вбудовані інтерфейси Python для служб операційних систем роблять його ідеальним засобом для написання утиліт адміністрування систем. Програми можуть проводити пошук в файлах, деревах каталогів, запускати інші програми, забезпечувати паралельну обробку за допомогою процесів та потоків і багато іншого.

Графічні користувацькі інтерфейси

- Простота та зручність розробки на Python також роблять його гарним інструментом для розробки настільних графічних користувацьких інтерфейсів.
- Все це забезпечують основні бібліотеки типу Tkinter, PyQT, KivyMD.

Написання веб-сценаріїв

- Говорячи про Python не можна не згадати про веб-програмування. Технологічні гіганти типу Google,YouTube, Netflix активно використовують Python для створення своїх веб-продуктів. Бібліотеки типу Flask, Django, FastAPI роблять його надпотужним інструментом для веб-програмування.

Машинне та глибоке навчання

- На сьогодні Python де-факто це стандарт для аналізу даних та машинного навчання. Його простота у синтаксі, швидкість у написанні

роблять надзручним інструментом для аналізу даних. А бібліотеки scikit-learn, PyTorch, Keras, Scipy, Numpy ще більше полегшують роботу. І це одна із основних причин, чому дана робота виконуєтьс саме на мові Python.

3.2 Бібліотека глибокого навчання Pytorch

Глибоке навчання дозволяє виконувати широкий спектр складних задач, наприклад, виконувати машинний переклад, грати в стратегічні ігрі, індентифікувати об’єкти на відео чи навіть керувати машиною. Тому, як правило, для такого спектру задач потрібен гнучкий інструмент, яким легко і просто користуватися на практиці. Також потрібно, щоб навченна мережа коректно працювала, якщо в даних існує певна невизначеність(що доволі часто буває).

PyTorch — сучасна бібліотека глибокого навчання, розробкою якої займається компанія Facebook.

Фреймворк глибокого навчання повинен вміти робити лише три речі:

- Визначити граф обчислення
- Диференціювати граф обчислення
- Обчислювати граф

Головна особливість Pytorch в тому, що цей фреймворк має динамічний граф обчислення. Це дуже потужний підхід, адже це дає можливість використовувати всі можливості мови програмування і не обмежує користувача ні в чому[8].

Навіть не дивлячись на те, що з самого початку PyTorch розроблявся лише для дослідницьких робіт на сьогоднішній день моделі створені на Python можна легко змігрувати на C++[20]. Це дивовижно, адже це дозволяє використовувати всю простоту і швидкість розробки на Python і водночас використовувати швидкість роботи програми на C++.

Ключові можливосты Pytorch

- Модель створенна на PyTorch відразу готова до “бойового середовища”
- Роздільне навчання
- Надійна екосистема бібліотеки
- Підтримка хмарних середовищ, що вкрай важливо із гігабайтами даних

3.3 База даних PostgreSQL

PostgreSQL — не просто реаліаціяна база даних, а об’єктно-реляційна СУБД. Це дає деякі переваги над іншими SQL базами даних з відкрим вихідним кодом, такими як MySQL, MariaDB та Firebird.

Фундаментальна характеристика об’єктно-реляційної бази даних — підтримка користувацьких об’єктів та їх поведіка, включаючи типи даних, функції, операції, домени та індекси. Це робить Postgres неймовірно надійним та гнучним. До того, він вміє зберігати та працювати із складними структурами даних[9].

PostgreSQL підтримує великий набір вбудованих типів даних:

- Числові типи
- Цілі
- Грошовий тип
- Символьні типи довільної довжини
- Типи «дата/час»
- І багато інших типів

Ключові особливості PostgreSQL

Безпека

- PostgreSQL дозволяє підключатися по захищенному SSL-з’єднанню і надає змогу аутентифікації по паролю та аутентифікацію за допомогою сторонніх сервісів(PAM, Kerberos)

Надійність та стабільність

- PostgreSQL дозволяє налаштовувати резервування, відновлення, а також різні види реплікацій

Підтримка транзакційності

- Для цього використовується механізм багатоверсійного керування одночасним доступом, який дозволяє не використовувати блокування рядків у всіх випадках, окрім зміни одного і того же рядка в різних процесах.

3.4 Бібліотека компютерного зору OpenCV

OpenCV — бібліотека компютерного зору із відкритим кодом. Вона написана на мові C++, але має API для Python, Matlab, Java, C++. Бібліотека спеціально розроблена для ефективного та швидкого користування. Саме тому, вона є де-факто стандартом для програм, які працюють в режимі реального часу. OpenCV включає в себе більше ніж 2500 тисячі оптимізованих алгоритмів, включаючи як алгоритми стандартного компютерного зору так і алгоритми машинного навчання[10]. Серед цих алгоритмів є такі що розпізнають обличчя, створюють 3D модель об’єкта із картинки, підвищують якість картинки і багато іншого. Використання цього продукту такими компаніями як Google, Toyota, Yahoo, Microsoft, Intel, IBM, Honda говорить саме за себе.

Бібліотеку OpenCV використовують як ключову складову при створенні роботів. У Китаї, Японії створенні системи за допомогою OpenCV для індентифікації людей.

Отже сумуючи все вище сказане, можна сказати що OpenCV — незамінна річ при створенні системи компютерного зору, яка працює в режимі реального часу.

3.5 Бібліотека графічного дизайну Kivy

Kivy — це безкоштовна бібліотека Python з відкритим вихідним кодом для розробки мобільних та настільних додатків.

Переваги

- Повністю працює на чистому Python, що робить розробку швидкою і зручною
- Код написаний переноситься один раз, переноситься на будь-яку платформу

Недоліки

- Недостатній рівень документації та прикладів

3.6 Інструмент контейниризації Docker

Сучасна розробка програми це не просто написання коду, це багаточисельні бібліотеки, фрейворки, архітектури і інші інструменти. Кожен із цих перечислених речей створюють просто величезну складність для майбутнього розвернення всієї системи. Саме через цю проблему з’явилися інструменти контейнеризації, а саме — Docker.

Docker container — стандартизована повноцінна одиниця програмного забезпечення, яка дозволяє розробникам ізолювати свою програму від зовнішнього світу і завдяки цьому зникає проблема перенесення програми на будь-яку платформу. Говорячи дуже просто і термінами ООП docker image — клас, а docker container — об’єкт класу. Тобто, це означає що можна створити безліч контейнерів з одного імаджу. Docker image складається із шарів, які накладені один на одного. Завдяки цьому ізолюються процеси, які знаходиться всереді імаджу[11].

Контейнери не потребують встановлення операційної системи, адже вони розділяють ядро операційної системи. Завдяки цьому збільшується ефективність роботи сервера і зменшуються затрати на цей самий сервер на якому розміщений контейнер. Програмні продукти вимагають безпеки, а контейнер може забезпечити цю безпеку на найвищому рівні.

Процес створення image доволі простий. Потрібно лише правильно описати, що саме має відбуватися із вашою програмою всередині контейнеру. Весь процес збірки контролюється файлом Dockerfile і/або

docker-compose.

Ще одною причиною використання Docker’у є хмарне сховище DockerHub. Це спеціальний веб-ресурс, який дозволяє завантажувати власні Docker imag’и, а також контролювати версії, доступи і багато чого іншого. Не можна не сказати, що docker imag’и можна використовувати при збіркі власних imag’ей. Це доволі частна практика, оскільки такий процес економить купу часу розробникам, а також зменшує ризик помилок при розробці.

3.7 Висновки до розділу

Оскільки, головним чином робота пов’язана із нейронними мережами, то було обгрунтовано використання бібліотеки глибокого навчання PyTorch і чому вона найкращим чином підходять для розв’язання задачі індентифікації аномальних подій на відео. До того ж було описано основні засоби розробки даної системи, такі як мова програмування Python, база даних PostgreSQL і їх переваги при розробці програмного забезпечення. Було представлено бібліотеку для створення графічного інтерфейсу користувача і вказано її основні переваги та недоліки. Також було обґрунтовано, що представлє собою засіб контейнеризації Docker і чому він найкраще підходить для швидкого та зручного розвернення програми на будь-якій платформі чи системі. Важливо зазначити, що завдяки вибраним інструментам можна легко розширяти та підримувати програмний продукт.

4. Опис програмної реалізації

4.1 Набір даних для задачі

На сьогоднішній день існує декілька датасетів для задачі розпізнавання аномалії на відео. Але, на жаль, майже всі вони не підходять для цієї задачі. Адже за метою цієї роботи є не тільки розпізнання дії, яка не схожа на інші, типу біг, паніка і т.д, а ще аномалії типу крадіжка, битва, вибух і т.д.

На рисунку 4.1 представлено порівняльну характеристику різних датасетів.

	Number of videos	Dataset lenght	Examples anomalies
UCSD Ped2	28	5 min	Walking across walkways, small carts
Avenue	37	30 min	Run, throw
UMN	5	5 min	Run
BOSS	12	27 min	Harass, Disease, Panic
UCSD Ped1	70	5 min	Walking across walkways, small carts
UCF-Crime	1663	128 hours	Abbuse, Arson, Assault, Fighting

Рисунок 4.1 — характеристика датасетів

Очевидно, з поміж усіх датасетів для цієї роботи найкраще підходить датасет під назвою UCF-Crime. Він знаходиться у вільному доступі, а також вміщає в себе 14 різних класів, як аномальних так і звичайних відео.

Загальна кількість годин у цьому датасеті — 128. Відео із камер відеоспростереження різних місць: вулиці, банки, торговельні центри, магазини, приватні будинки, перехрестя та ін.

В даній роботі всі аномальні події об’єднано в один клас, а звичайні в інший. Мотивація цього рішення наступна: для мультикласової класифікації потрібно набагато більше відео із аномальними подіями. Але до задачі

мультикласової можна підійти з іншого боку. Якщо базова модель визначить, що дійсно сегмент відео належить до аномального класу, то маючи цю інформацію можна натренувати іншу модель, яка буде визначити конкретну аномальну подію.

Для розробки системи мультикласової класифікації потрібні великі обчислювальні ресурси. Тому рішення було прийнятно на користь бінарного класифікатора. Повний перелік доступних класів із відповідною кількістю відео у датасеті UCF-Crime представлено на Рисунку 4.2.

Class Name	Number of videos
Abuse	50
Arrest	50
Arson	50
Assault	50
Burglary	100
Explosion	50
Fighting	45
Road Aciddents	127
Robbery	145
Shooting	27
Shoplifting	29
Stealing	95
Vandalism	45
Normal event	800

Рисунок 4.2 — доступні класи в датасеті UCF-Crime

Як було зазначено, аномальна подія — це подія, яка як правило яка займає маленький сегмент у часі, а отже нейронна мережа має оцінювати маленькі проміжки часу. Наступний графік на рисунку 4.3 це підтверджує, приблизно 1300 відео мають часовий проміжок який менше ніж 3 хвилин і лише 363 більше ніж 5 хвилин.

На рисунках 4.4 — 4.10 зображено різні приклади відео, на яких відбувається тренування моделі.

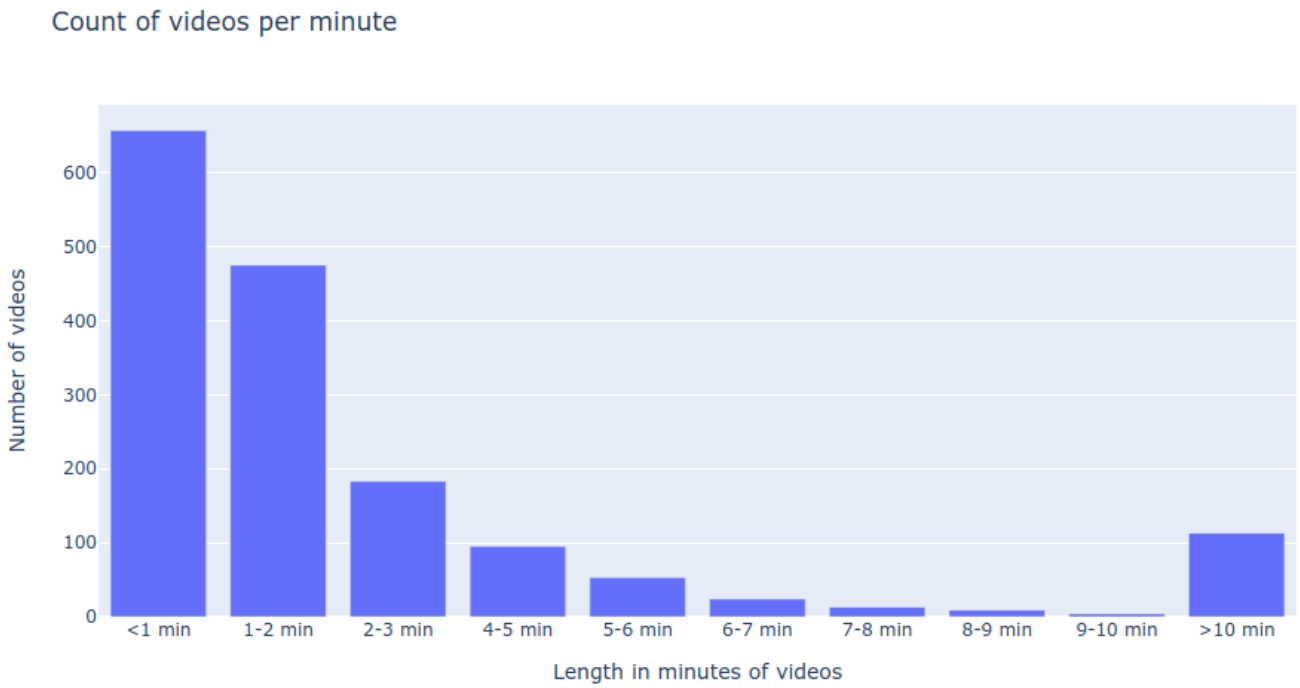

Рисунок 4.3 — розподіл відео по часу

На рисунку 4.4 зображено приклад аномального відео — крадіжки зі зломом.

Рисунок 4.4 — крадіжка зі зломом

На наступних рисунках 4.5-4.6 зображено приклади подій, що відмічені як

звичайні, тобто не мають кардів аномальних подій, таких як крадіжка, бійка, дорожно-транспортна пригода, вбивство, підпал.

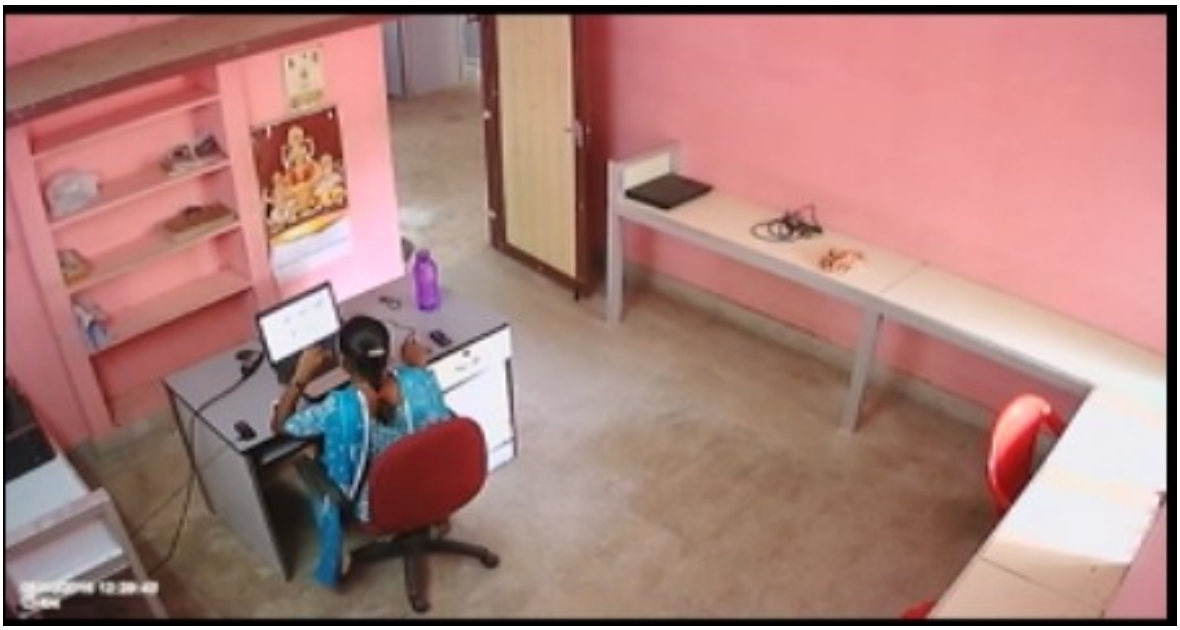

Рисунок 4.5 — приклад звичайного відео

Іноді зустрічаються події, які складно роздивитися на камері (рисунок 4.8).

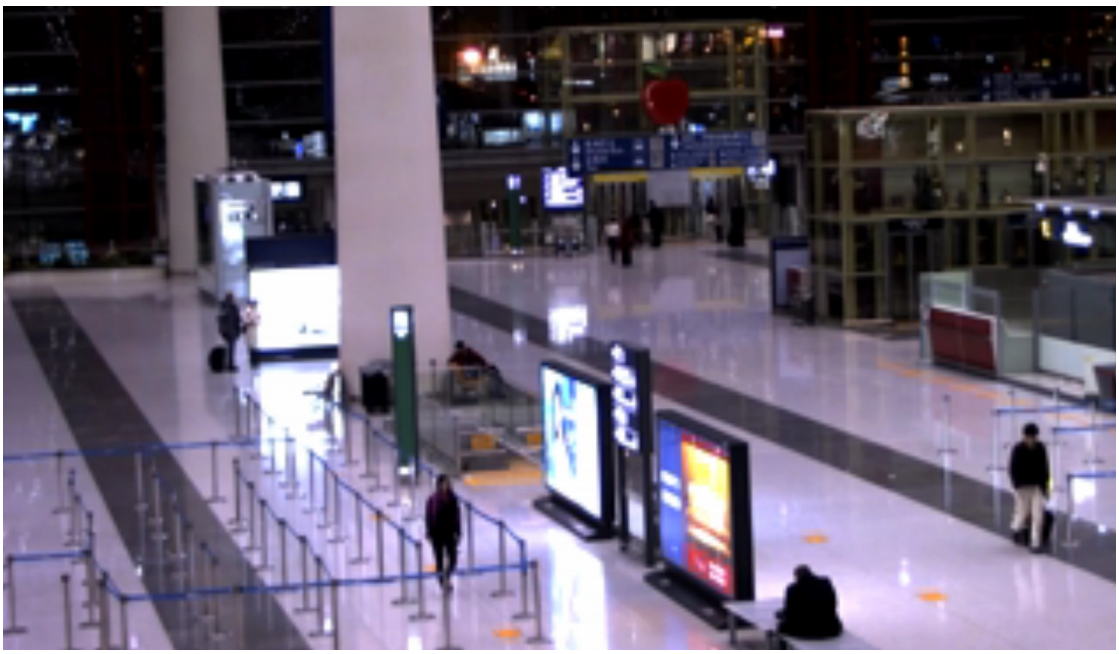

Рисунок 4.6 — приклад звичайного відео

Але навіть їх модель зможе розпізнати , адже відео поганої якості часто зустрічаються у тренувальному наборі.

На риунку 4.7 зображено приклад нападу.

Рисунок 4.7 — приклад нападу

На рисунку 4.8 представлено приклад знущання.

Рисунок 4.8 — приклад знущання

На рисунку 4.9 зображено приклад насильства — чоловік б’є собаку через огорожу.

Події на кадрах 4.9 — 4.10 важко розпізнати подію навіть для людського ока.

Рисунок 4.9 — приклад насильства

На рисунку 4.10 зображено приклад вибуху у приміщенні.

Рисунок 4.10 — приклад вибуху

Різноманітніість вмісту і якості зображень забезпечить високу якість моделі.

4.2 Імплементація алгоритму

Так як датасет не потребував певної очистки(типу видалення відео, які не підходять під відповідний клас), як це часто буває в задачах машинного навчання, то реалізація цього алгоритму почалася із так званої попередньої обробки відео. На тренувальному датасеті було пораховано статистики для середнього для всіх 3 каналів RGB. Кожен фрейм із відео було нормалізовано, а також зменшено до 240x320 пікселів. Ця обробка потрібна, із двох причин. Перша, на вхід нейронній мережі потрібно завжди подавати дані одного масштабу для коректної роботи алгоритму градієнтного спуску (Рисунок 4.11) [19].

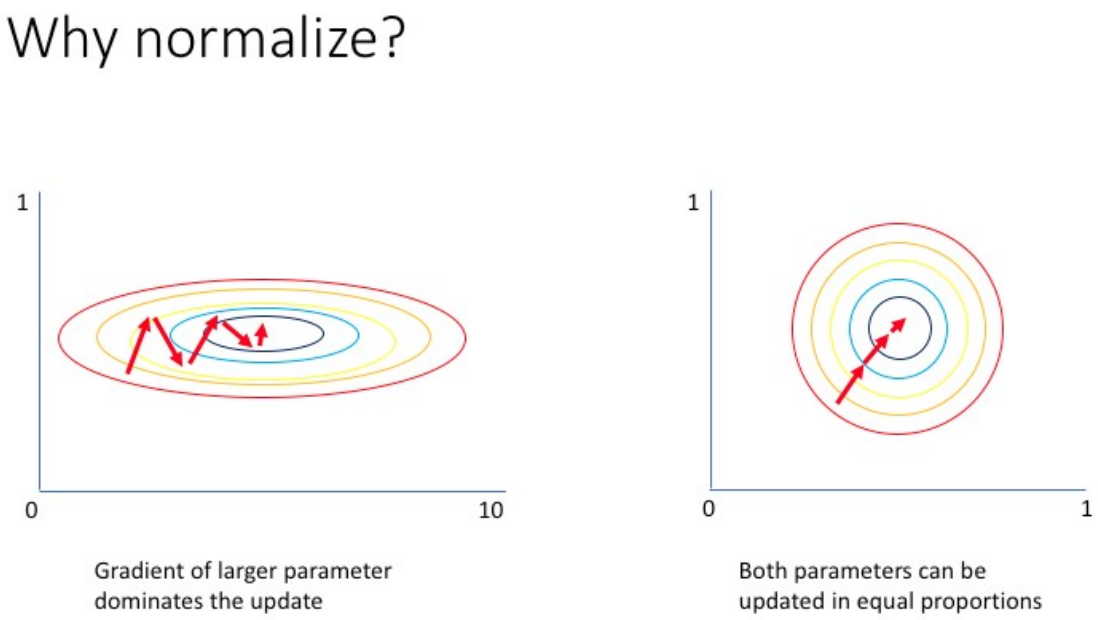

Рисунок 4.11 — лінії рівня

Уявімо ситуацію в якій є дуже проста нейронна мережа із двома входами. Перший параметр x1 масштаб якого варіється від 0 до 1, а другий параметр x2 від 10 до 1000. Так як задача мережі навчиться комбінувати ці входи за допомогою лінійних операцій та нелінійних активацій, то такий випадок може “надурити” градієнтний спуск, оскільки алгоритм оптимізації замість

того, щоб знайти оптимальне рішення буде намагатися вирівняти масштаби ознак.

Тому масштабування це дуже важливо з точки зору оптимізації, а також важливо з точки зору швидкості роботи. Оскільки алгоритм буде лише зосереджений на пошуку оптимального рішення, то він зійдеться набагато швидше. Друга причина такої попередньої оброки це збільшення швидкості при навчанні завдяки зменшенню розмірності картинки, а відповідно даних які треба мережі обробити.

За допомогою моделі C3D (Рисунок 4.12), яка навчена на відомому датасеті Kinetics було визначено ознаковий опис кожного відео.

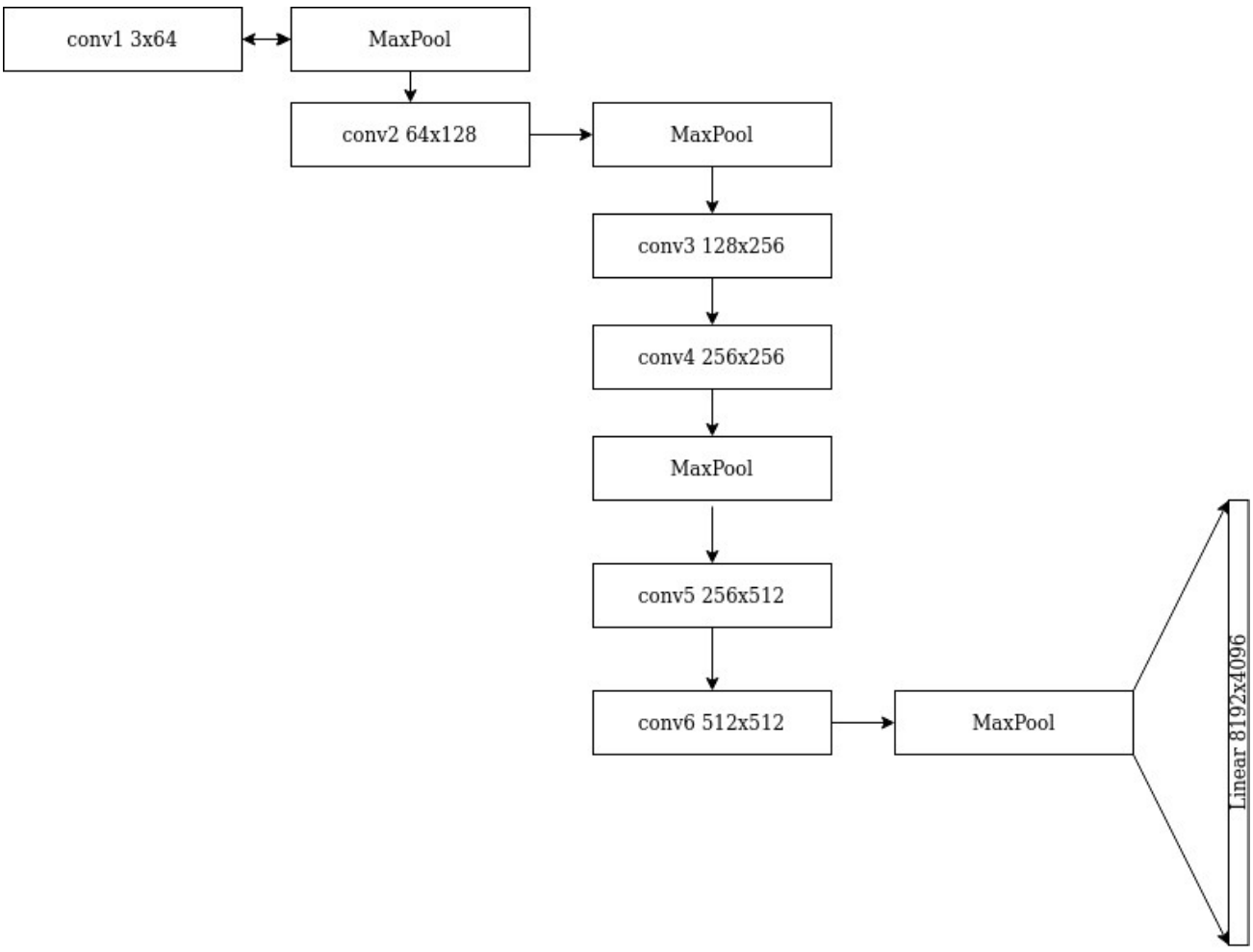

Рисунок 4.12 — архітектура моделі C3D

C3D або Convolution 3D — глибока 3-х мірна конволюційна нейронна мережу із однорідною архітектурою, яка складається із 3x3x3

конволюційного ядра та пулінга 2x2x2 після кожного шару. Вперше ця архітектура була представлена світу у 2015 році. Така архітектура вдало підходить для даних, які мають як просторову структуру, так і часову компоненту пов’язану із рухом об’єктів, людськими діями. Саме тому системи, які використовують цю архітектуру займають переможні та призові місця для різних задач пов’язаних із комп’ютерним зором на платформі Kaggle. Важливо зазначити, що завдяки однорідній архітектурі та невеликими розмірами ядер(ваги) модель працює набагато швидше у порівнняні із такими архітектурами як ResNet, AlexNet та інші. Важливою особливістю також є те, що на виході цієї мережі ми отримуємо компактне представлення сегменту відео із N кадрів у вигляді вектору розмірності 4096. Завдяки цьому можна отримувати векторне представлення певного сегменту, а як вже сказано це дуже важливо в задачі розпізнання аномалій, оскільки потрібно оцінювати невеликі сегменти часу.

Провевши попередню обробку відео було створено послідовності із N(для кожного відео різне) кадрів після чого сегменти було подано на вхід C3D моделі. Результатом такої роботи стала матриця A розмірності Nx4096. Число 4096 — вихід моделі C3D.

Маючи матрицю A(різна для кожного відео) за допомогою інтерполяції було усереднено відео до розмірності 32x4096. По-перше, розмірність входу класифікатора має бути статичною. По-друге, це дає змогу зекономити на пам’яті(вкрай важливо!).

Наступним кроком в реалізації даної системи було створення класифікатора. Початкова ініціалізація вагів була проведена за схемою Хав’єра[2]. Також в класифікаторі я використовую новий шар — Dropout. Головна ціль цього шара — зменшити перенавчання за рахунок “забуття” певного відсотка вагів після кожної епохи навчання. Я експерементував із мережам із більшою кількістю прихованих шарів, але це не дає ніяких результатів. Головна причина — затухання градієнту. Затухання градієнту робить неможливим зміну вагів, а отже і навчання.

Також для оптимізації я використовував AdaGrad(різновидність градієнтного спуску).

Отже результатом роботи цієї мережі стане вектор розмірності 32. Кожне із цих 32-х значень є ймовірністю того, що сегмент(який в свою чергу складається із 16 кадрів) є аномалією[12]. (Рисунок 4.13)

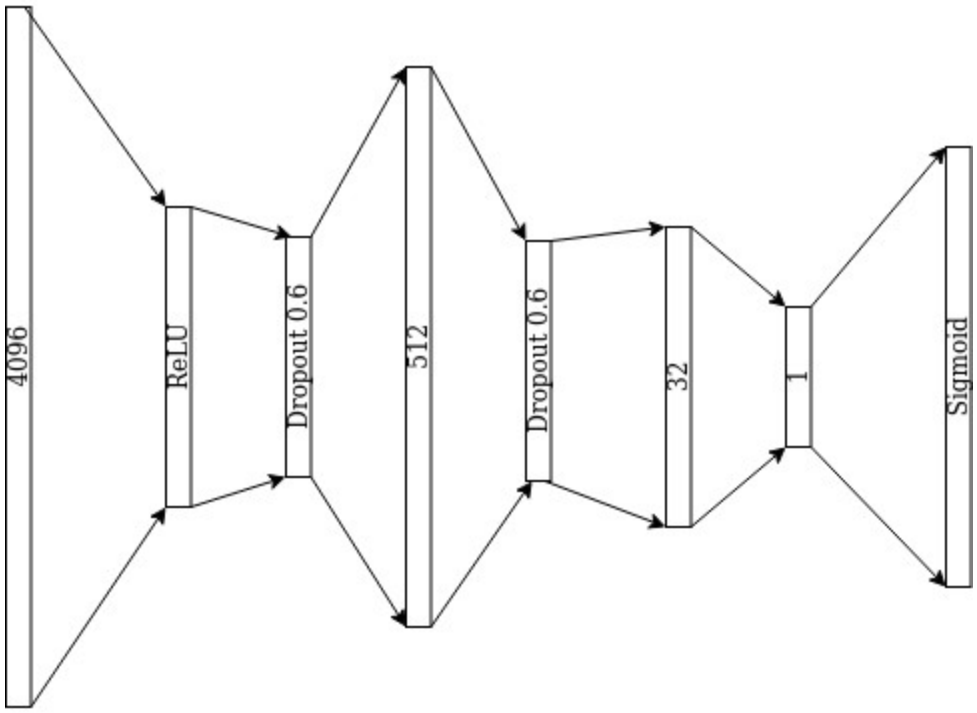

Рисунок 4.13 — архітектура класифікатора

Це дає деяку варіантивність в тому, що кінцеву відповідь аномалія або звичайне відео можна отримати декількома способами:

- усереднити
- взяти максимальне значення
- кожен із сегментів вважати незалежною одиницею

Для тренування також було встановлено батч із 60 відео. 30 відео відносяться до позитивного класу, а інші 30 — негативного.

Функцією помилки було обрано MIL Ranking Loss[13], оптимізуючи яку я очікую, що сегменти аномалії отримають більше значення ніж сегменти звичайного відео. Можна розглядати цю функцію помилки як для задачі

регресії.

Для перевірки якості алгоритму як під час тренування, так і під час тестування обрано ROC-AUC метрику, оскільки просте accuracy буде нечесно використовувати для задачі із незбалансованою вибіркою.

Сумуючи все вище сказане схему можна представити у вигляні рисунка 4.14.

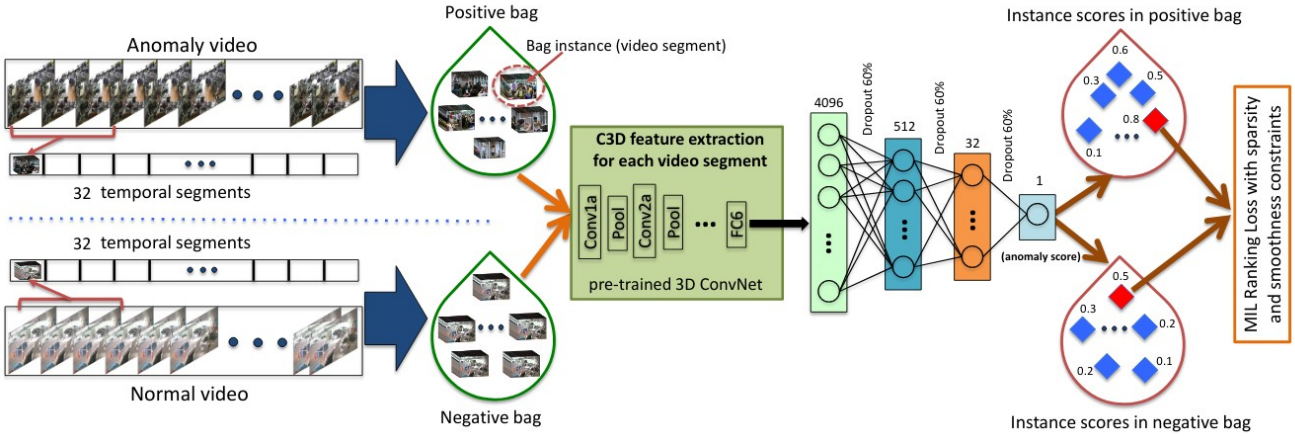

Рисунок 4.14 — Схематичне представлення системи[11]

Важливою частиною процесу навчання є кількість епох та швидкість навчання(learning rate). Для даної задачі Epochs = 20000. Learning_rate = 0.01. Аналізуючи графік на рисунку 4.15, приходимо до висновку що процес навчання моделі був успішній, без перенавчання.

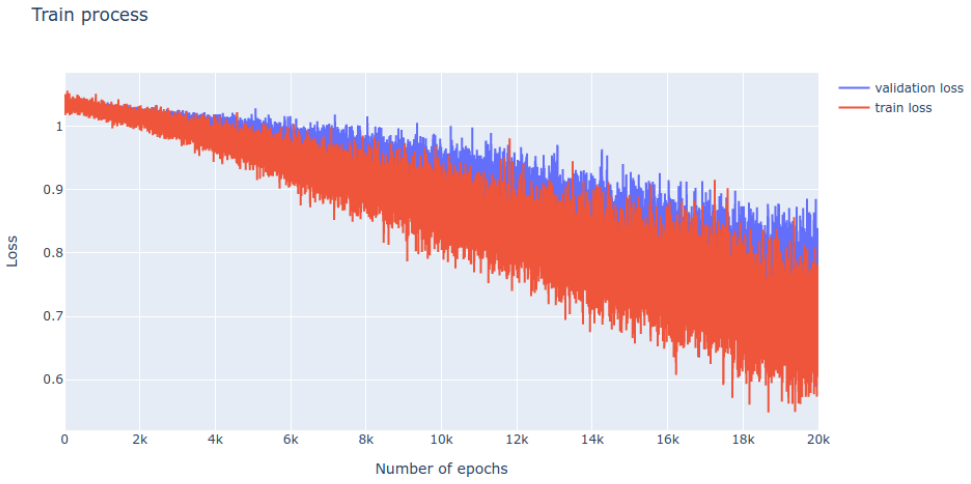

Рисунок 4.15 — графік навчання моделі

Також в цьому можна переконатися дивлячись на значеннях метрик.

Значення ROC-AUC для тренувальної вибірки становить 0.74 (Рисунок 4.16).

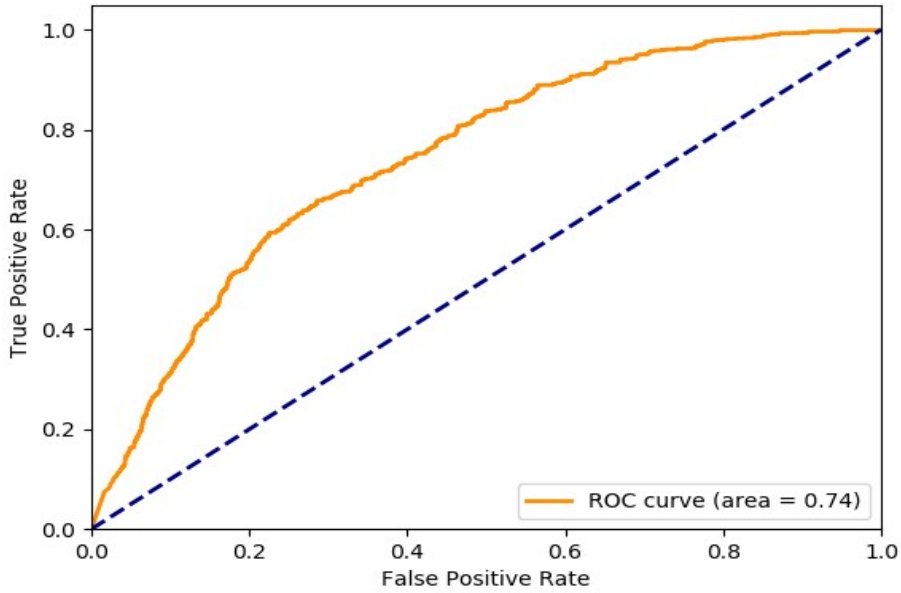

Рисунок 4.16 — графік ROC-AUC для трейна

Значення ROC-AUC для тестової вибірки становить 0.73 (Рисунок 4.17).

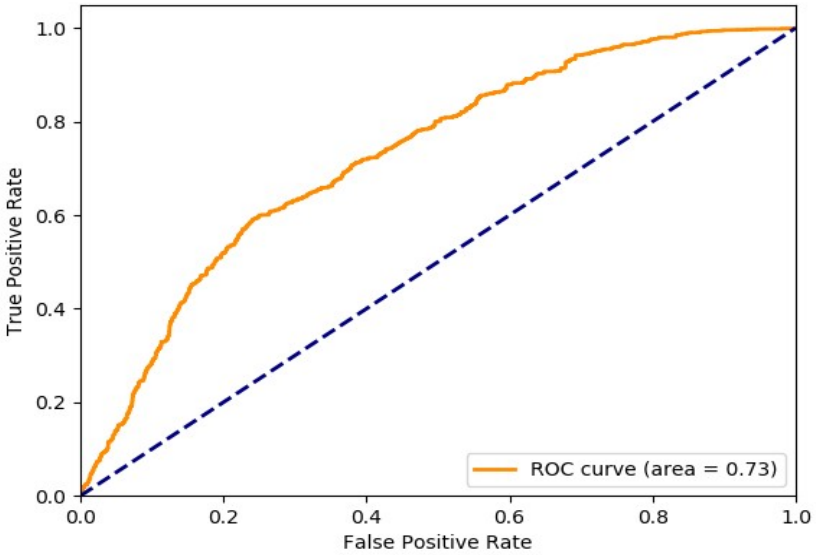

Рисунок 4.17 — графік ROC-AUC для теста

Порівнюючи значення метрики як на тренувальній, так і на тестовій можна сказати що модель вивчила патерни і не перенавчилася.

4.3 Опис таблиць бази даних

Для реалізації бази даних було створено дві таблиці (Рисунок 4.18):

- video
- video description

Таблиця video — це батьківська таблиця, яка має лише колонку “назва” та “шлях до відео”.

Інша таблиця video_description — має назву відео, початок та кінець сегменту відео та відповідна мітка до цього сегменту.

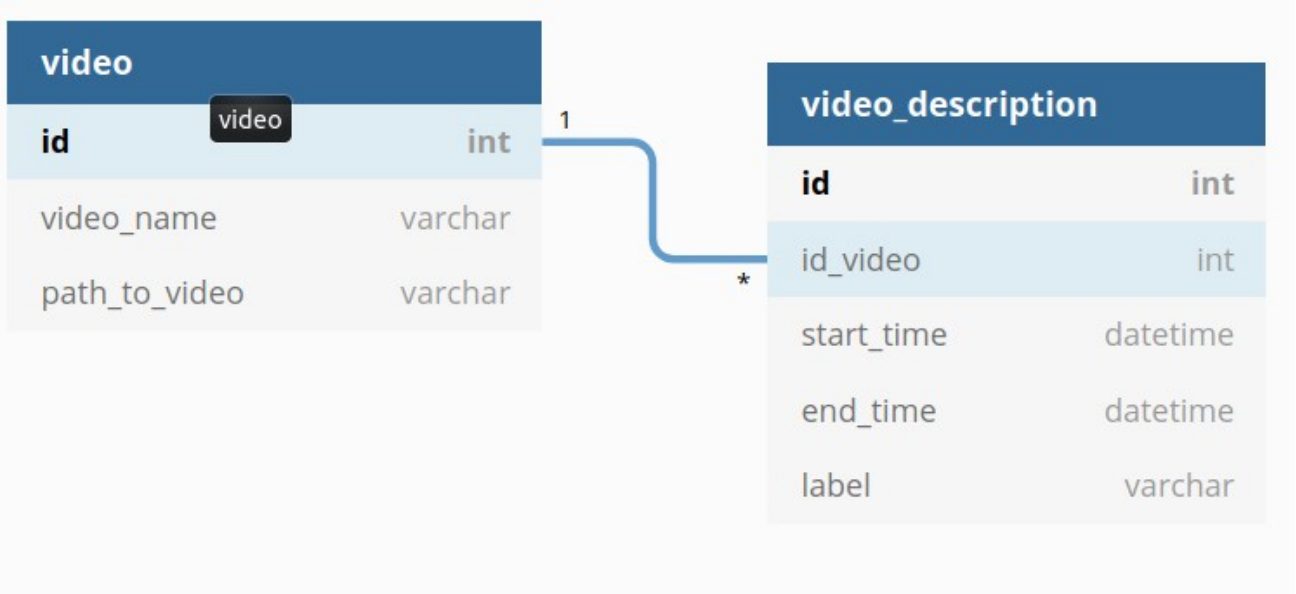

Рисунок 4.18 — структура бази даних

Таблиця video:

- id — первинний ключ
- video_name — назва відео
- path_to_video — повний шлях до файла

Таблиця video_description:

- id — первинний ключ
- id_video — зовнішній ключ

• start_time — початок події

• end_time — кінець події

• label — мітка часового вікна

Зовнішній ключ типу int id_video дозволяє зв’язати таблицю video_description та video. У цих двох таблиць зв’язок один(video) до багатьох(video_description). Логіка в цьому наступна: так як таблиця video — батьківська, то кожен запис в цій таблиці — унікальний день. Протягом дня у будь-який момент часу може статися чи то подія аномальна, чи то бути звичайна, повсякденна подія. Тому у таблиці video_description може бути декілька записів із відповідною міткою і часовим вікном для одного відео. Запис додається лише тоді, коли змінюється подія.

Також для таблиці video_description стоїть налаштування ON UPDATE: RESTRICT, що означає неможливе вставлення запису в базу, якщо запису не існує у батьківській таблиці. Цей механізм повністю забезпечує нам цілісність бази даних.

Цілостність бази даних — властивість бази даних, яке означає, що БД містить несуперечливу і повну інформація необхідну для коректного функціонування системи. Відповідно для забезпечення цілісності БД накладають деякі обмеження цілісності. Окрім RESTRICT існує CASCADE, SET NULL, NO ACTION.

У майбутній версії цього продукту модель бази даних буде значно розширена для нового функціоналу.

4.4 Процес збірки додатку за допомогою Docker

Процес збірки програмного продукту за допомогою інструменту контейниризації зводиться до написання спеціальних файлів Dockerfile, в якому описується які команди потрібно виконати при збіркі. А також файл docker-compose в якому описується залежність між різними мікросервісами в контейнері.

В моєму випадку двома мікросервісами виступають база даних, а також сам додаток. На рисунку 4.19 представлено файл Dockerfile. Ключова команда FROM означає, що в image який створюється потрібно додатково додати вже існуючий image. В моєму випадку я використовую image — postgresql.

```
FROM library/postgres

RUN apt-get update && apt-get install -y python3 python3-pip
RUN apt-get install python3-dev -y

RUN apt-get install libgl1-mesa-glx -y
RUN apt-get install libmtdev1 -y
RUN apt-get install -y libsm6 libxext6 libxrender-dev -y
RUN apt-get install libxrender1 -y


RUN python3 -m pip install --upgrade --user pip setuptools virtualenv
RUN python3 -m pip install ffpyplayer
RUN python3 -m pip install kivy

COPY ./ ./

WORKDIR ./

RUN pip3 install -r requirements.txt
```

Рисунок 4.19 — вміст Dockerfile

Як було сказано, файл docker-compose створений для того, щоб контролювати мікросервіси.

```
version: '3'
services:
  app:
    build: .
    command: ["python3", "./main.py"]
    depends_on:
      - "db"
    environment:
      DISPLAY: unix$DISPLAY
    volumes:
      - /tmp/.X11-unix:/tmp/.X11-unix
  db:
    image: postgres
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: rbhgbljy1
      POSTGRES_DB: criminal_video_detector
```

Рисунок 4.20 — вміст docker-compose

В моєму випадку це мікросервіс db — база даних, а також app — gui додаток

разом із нейроннож мережею.

4.5 Висновки до розділу

Це основний розділ в якому описано як саме створювався програмний продукт за темою аналіз відеопотоку: індентифікація аномальних подій на відео в режимі онлайн. Обгрунтовано вибір датасету UCF-Crime серед всіх доступних датасетів у мережі інтернет. Було обгрунтовано вибір преднавченої нейронної мережі Convolution 3D, а також архітектуру остаточного класифікатора. Описано процес попередньої обробки даних. Було описано процес створення вибірки для навчання, а також обгрунтовано необхідність аугментації даних та процес навчання моделі. Були представлені результати навченої моделі, а саме показано метрики, а також процес навчання моделі і встановлено, що нейронна мережа не перенавчилася і спроможна до практичного використання.

Останнім етапом було описано структуру бази даних, також було обгрунтовано використання саме такої структури. А одне із найважливіших моментів це те, що програма та база даних створена таким чином, щоб її легко можна було розширити. Для прикладу, можна створити систему таких програм, які керуються одним сервером і при необхідності відразу повідомлять про аномальну подію у відповідні органи безпеки і т.д.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

5.1 Системні вимоги та додаткове програмне забезпечення

Для якісної та безперервної роботи системи необхідні наступні характеристики ПК:

- Процесор Intel Core i7
- 8 Гб оперативної пам’яті
- Nvidia GeForce gtx 1050

Додатково необхідно мати встановлену відеокамеру та програмне забезпечення Docker. Відеопроцесор потрібен для швидкої роботи нейронної мережі, яка є ключом до всієї системи. Без відеопроесора будуть значні навантаження на CPU, а як наслідок робота із затримкою.

Нижче описано послідовність дій, необхідних для встановлення image та запуску програми(Приводиться приклад на базі системи Ubuntu 18.04).

1. Відкриваємо консоль(ctrl+alt+T) і вводимо команду «sudo docker-compose up». Результатом має бути наступний вивід логів(див.рисунок 5.1)

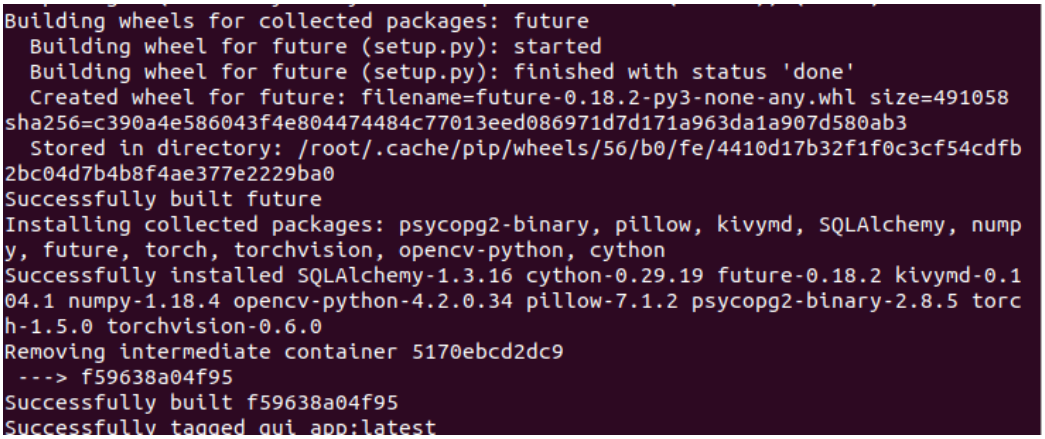

Рисунок 5.1 — логи після успішного збору

Видно, що програмний продукт успішно зібрано і готово до запуску.

2. Другим етапом є запуск контейнера (Рисунок 5.2), відбувається автоматично після збірки.

```
Attaching to gui_db_1, gui_app_1
app_1  | [WARNING] [Config      ] Older configuration version detected (0 instea
d of 21)
app_1  | [WARNING] [Config      ] Upgrading configuration in progress.
app_1  | [INFO    ] [Logger      ] Record log in /root/.kivy/logs/kivy_20-05-31_0
.txt
app_1  | [INFO    ] [Kivy        ] v1.11.1
app_1  | [INFO    ] [Kivy        ] Installed at "/usr/local/lib/python3.7/dist-pa
ckages/kivy/__init__.py"
app_1  | [INFO    ] [Python      ] v3.7.3 (default, Dec 20 2019, 18:57:59)
app_1  | [GCC 8.3.0]
db_1   |
db_1   | PostgreSQL Database directory appears to contain a database; Skipping i
nitialization
db_1   |
db_1   | 2020-05-31 14:44:36.918 UTC [1] LOG:  starting PostgreSQL 12.3 (Debian
12.3-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0
, 64-bit
db_1   | 2020-05-31 14:44:36.918 UTC [1] LOG:  listening on IPv4 address "0.0.0.
```

Рисунок 5.2 — логи запуску додатку

В результаті маємо успішний запуск системи із усіма необхідними компонентами для неї.

5.2 Результати виконання програми

На рисунку 5.3 представлено діаграму прецедентів програмного продукту.

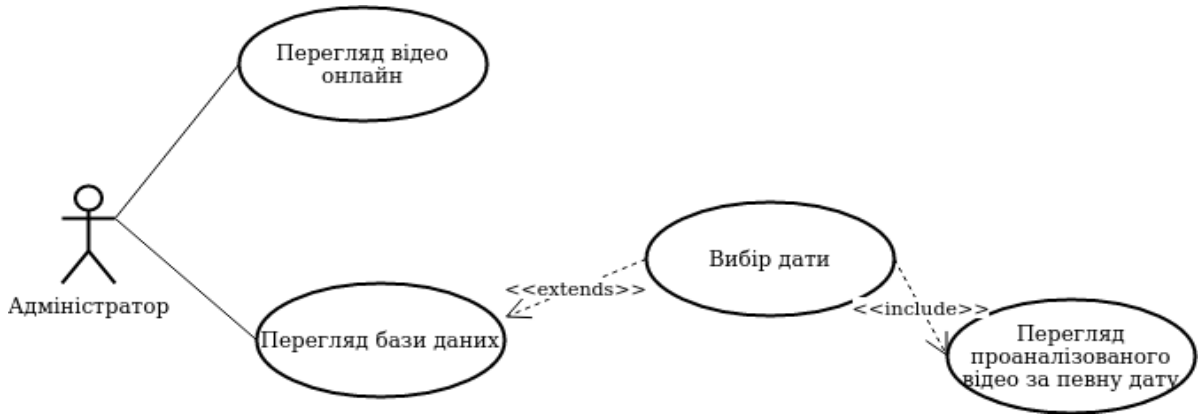

Рисунок 5.3 — діаграма прецедентів

У подальшому функціонал можна легко розширяти.

На рисунку 5.4 представлено головне вікно.

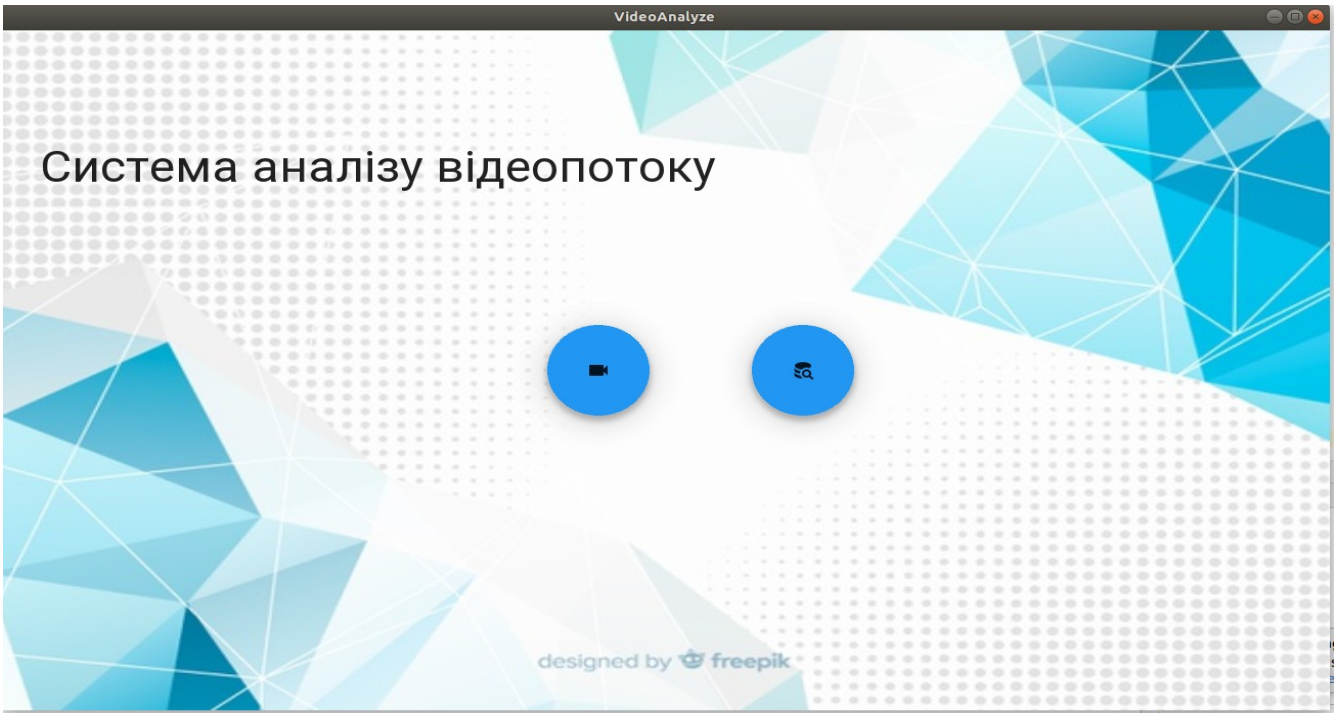

Рисунок 5.4 — Вікно меню

На головній сторінкі системи дві кнопки: перехід до камери та перегляд бази даних. На Рисунку 5.5 зображено вікно вибору дати.

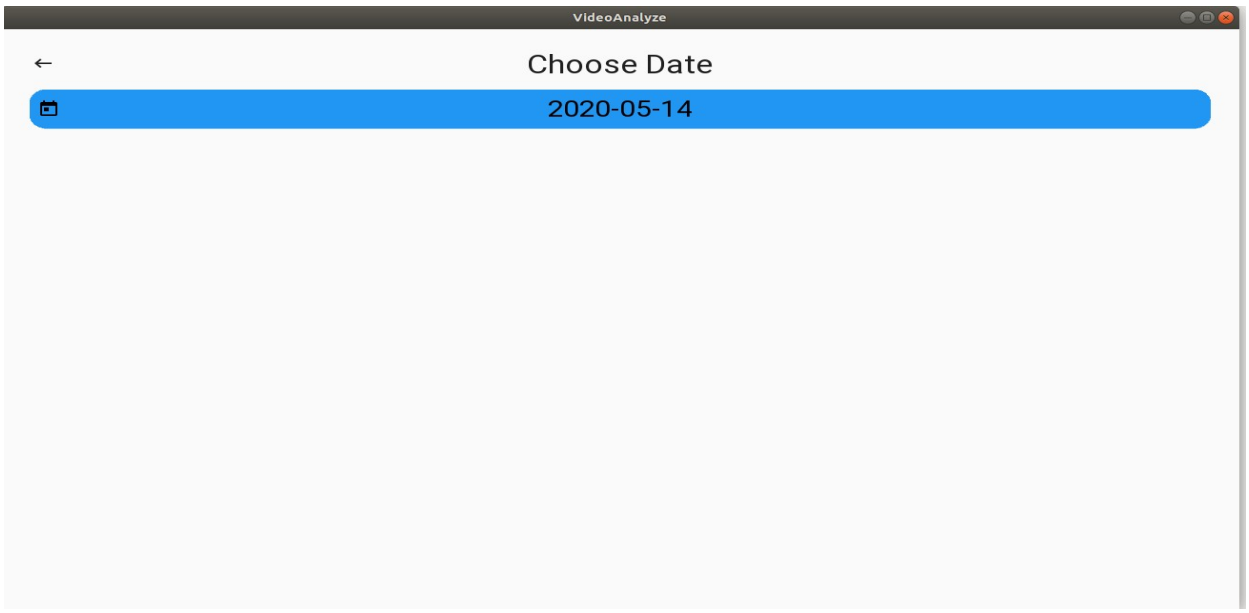

Рисунок 5.5 — Вікно вибору дати

Дата в цьому вікні унікальна і відповідає відео за конкретний день. Після вибору дати користувач попадає у вікно перегляду відеоспостережень за цю добу з аналізом відео (Рисунок 5.6).

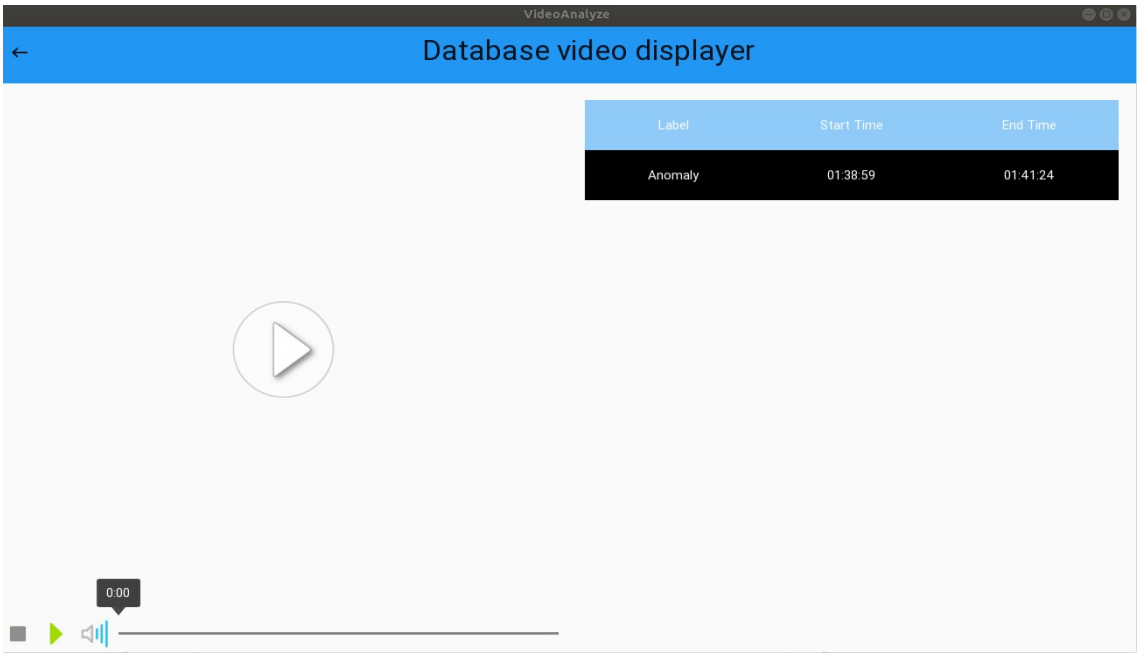

Рисунок 5.6 — Вікно перегляду проаналізованого відео

На рисунку 5.7 бійка і мережа класифікує цю подію аномальною.

Рисунок 5.7 — Приклад роботи нейронної мережі

На рисунку 5.8 видно, що чоловік справа штовхнув іншого чоловіка і мережа класифікує правильно подію.

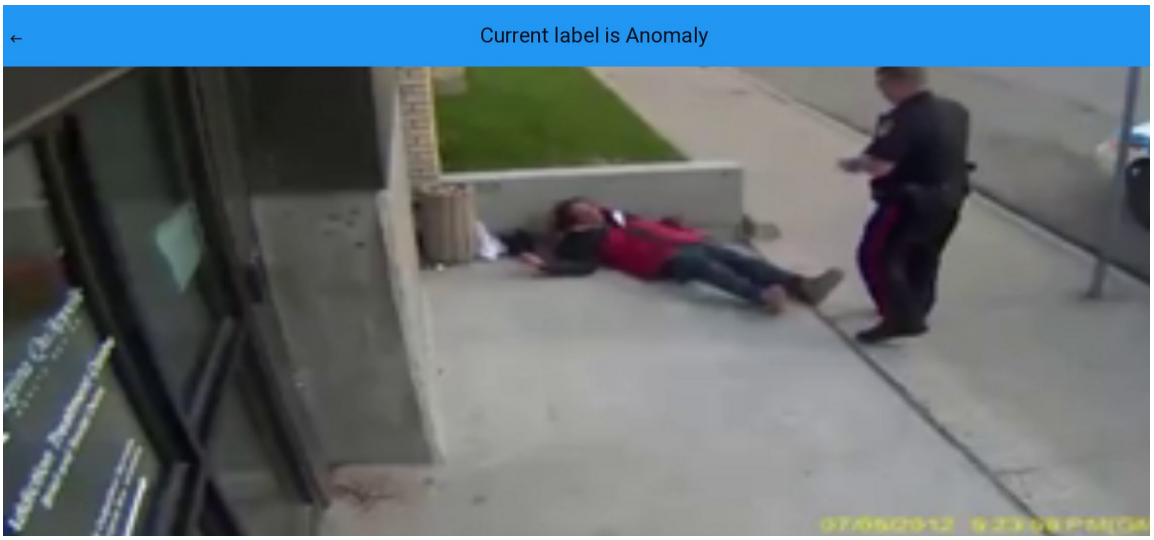

Рисунок 5.8 — Приклад роботи нейронної мережі

На рисунку 5.9 зображено приклад звичайної події.

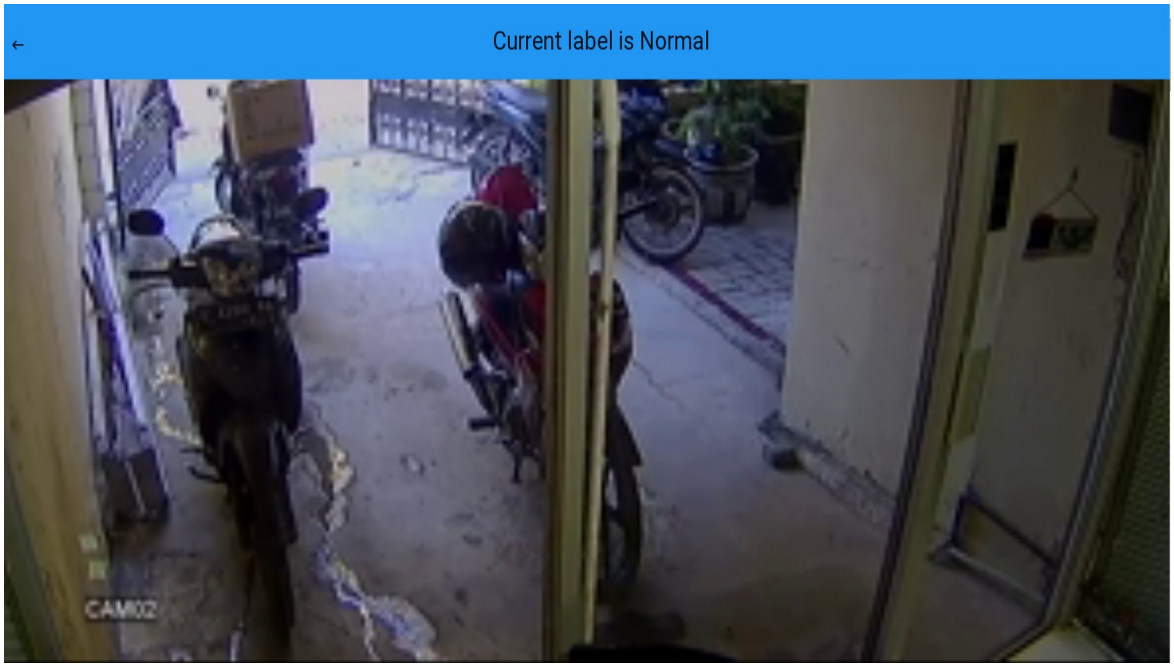

Рисунок 5.9 — Приклад роботи нейронної мережі

У майбутній версії повідовлятиметься ймовірність аномальної події.

5.3 Висновки до розділу

В даному розділі було описано необхідні системні вимоги для коректної та безперебійної роботи програми, розробленої в межах виконання дипломної роботи за темою „аналіз відеопотоку: ідентифікація аномальних подій“. Були вказані вимоги до апаратного забезпечення, на якому буде запускатися система аналізу і вимоги до програмного забезпечення, що необхідно для запуску і коректної роботи додатка.

Було описано процес запуску системи за допомогою Docker і проілюстровано варіант успішного виконання збору системи і запуску контейнера з усіма необхідніми для цього командами.

Було зображено функціонал продукту у вигляді діаграми прецедентів, після чого детально описаний графічний інтерфейс користувача, проілюстровані усі вікна додатка і описано їх значення в системі.

Висновки

В ході виконання даної роботи було проведено ретельний аналіз предметної області, включаючи аналіз вимог, вибір технологій, архітектуру системи, організацію програмного коду. Результат – було створено першу версію інформаційної системи аналізу відеопотоку в режимі онлайн. Закладено фундамент для подальшого вдосконалення системи: розробка централізованого серверу, який керуватиме не тільки однією камерою, а і всіма іншими. Також можна додати до функціоналу автоматичне повідомлення у відповідні органи про кримінальну подію із усіма супроводжуючими даними: відео сегмент, місце знаходження та рівень небезпеки.

Практичне значення результатів полягає у тому, що кринімальні події можна зменшити в рази. Як наслідок, на наших вулицях стане безпечніше для всіх нас. А комусь навіть це збереже життя.

Перелік використаних джерел

1. Leibe B. Computer Vision – ECCV 2016(Angry Crowds: Detecting Violent Events in Videos) / B. Leibe, J. Matas, N. Sebe. – Amsterdam, 2016.

2. History of neural nets [Електронний ресурс] // stanford – Режим доступу до ресурсу: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html.

3. Шитиков В.К., Розенберг Г.С., Зинченко Т. Д. Методы системной идентификации / Тольятти: ИЭВБ РАН. 2003. 463 с.

4. Будова нейрона [Електронний ресурс] // Хмельницький національний університет – Режим доступу до ресурсу: http://dn.khnu.km.ua/dn/k_default.aspx?M=k1113&T=05&lng=1&st=0.

5. How the backpropagation algorithm works [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: http://neuralnetworksanddeeplearning.com/chap2.html.

6. Goodfellow I. Deep Learning(c.326-335) / I. Goodfellow, B. Yoshua, A. Courville., 2016.

7. Лутц М. Изучаем Python. 5-е изд. Том 1(с.38-50) / Марк Лутц.. – 833 с.

8. Stevens E. Deep Learning with Pytorch(c.10-26) / E. Stevens, L. Antiga.. – 141 с.

9. Моргунов Е. П. Postgresql основы языка SQL / Е. П. Моргунов. – СПб, 2018.

10. Brashmbhatt S. Practical OpenCV / Samarth Brashmbhatt.

11. Turnbull J. The Docker Book(c.5-10) / James Turnbull., 2019.

12. Real-world Anomaly Detection in Surveillance Videos [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1801.04264.pdf#cite.IDM_AL.

13. Understanding Ranking Loss, Contrastive Loss, Margin Loss, Triplet Loss, Hinge Loss and all those confusing names [Електронний ресурс] – Режим доступу до ресурсу: https://gombru.github.io/2019/04/03/ranking_loss/.

14. Aurelien G. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems / Geron

Aurelien.

15. Bishop C. Pattern Recognition and Machine Learning / Cristopher Bishop., 2006.

16. Моуэт Э. Использование Docker / Эдриен Моуэт., 2017.

17. Рашид Т. Make your own neural network / Тарик Рашид., 2019.

18. Рашка С. Python и машинное обучение / Себастьян Рашка., 2019.

19. Normalizing your data [Електронний ресурс] – Режим доступу до ресурсу: https://www.jeremyjordan.me/batch-normalization/.

20. Pytorch Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://pytorch.org/docs/stable/index.html.

ДОДАТОК 1

Аналіз відеопотоку: ідентифікація аномальних подій

Специфікація

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61186_20Б

Аркушів 2

Київ — 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ ТМ61186_20Б 81-1	Павленко М.Р_ТМ61.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ ТМ61186_20Б 12-1	Inference.py	Модуль для роботи із нейронною мережею
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ ТМ61186_20Б 12-2	Dbrequests.py	Модуль для роботи із базою даних
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ ТМ61186_20Б 12-3	Windows.py	Модуль для роботи із вікнами графічного інтерфейсу

ДОДАТОК 2

Аналіз відеопотоку: ідентифікація аномальних подій

Текст програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61186_20Б 12-1

Аркушів 9

Київ — 2020

```
import numpy as  np
import pandas as pd
import os
from torch.utils.data import Dataset
from sklearn.model_selection import train_test_split
class FeaturesDataset(Dataset):
    def __init__(self, fnames, labels, sequence_len=32):
        self.sequence_len = sequence_len
        self.labels = labels
        self.fnames = fnames



    @staticmethod
    def interpolate(features, features_per_bag):
        feature_size = features.shape[1]
        interpolated_features = np.zeros((features_per_bag, feature_size))
        interpolation_indicies = np.round(np.linspace(0, len(features) - 1,
num=features_per_bag + 1))
        count = 0
        for index in range(0, len(interpolation_indicies)-1):
            start = int(interpolation_indicies[index])
            end = int(interpolation_indicies[index + 1])

            assert end >= start

            if start == end:
                temp_vect = features[start, :]
            else:
                temp_vect = np.mean(features[start:end+1, :], axis=0)
```

```
            temp_vect = temp_vect / np.linalg.norm(temp_vect)

            if np.linalg.norm(temp_vect) == 0:
                print("Error")

            interpolated_features[count,:]=temp_vect
            count = count + 1

        return np.array(interpolated_features)

    def __len__(self):
        return 60

    def get_features(self):
        if self.state == 'Normal':
            index = np.random.choice(np.where(self.labels == 0)[0])
        elif self.state == 'Anomalous':
            index = np.random.choice(np.where(self.labels == 1)[0])

        path_to_features = self.fnames[index]
        data = np.loadtxt(path_to_features)

        if data.shape[0] != self.sequence_len:
            data = self.interpolate(data, self.sequence_len)

        features = self.interpolate(data, self.features_per_bag)

        if self.state == 'Normal':
            self.state = 'Anomalous'
        elif self.state == 'Anomalous':
```

```
            self.state = 'Normal'

    def __getitem__(self, index):
        features, label = self.get_features()
        return features, label
result = []
for label in os.listdir('out'):
    for video in os.listdir(f'out/{label}'):
        item = {'video': f'out/{label}/{video}',
                'label': label}
        result.append(item)

frame_height = 240
frame_width = 320
channels = 3
frame_count = 16
features_per_bag = 32

def sliding_window(arr, size, stride):
    num_chunks = int((len(arr) - size) / stride) + 2
    result = []
    for i in range(0,  num_chunks * stride, stride):
        if len(arr[i:i + size]) > 0:
            result.append(arr[i:i + size])
    return np.array(result)

def chunks(l, n):
    for i in range(0, len(l), n):
        yield l[i:i + n]
```

```
def interpolate(features, features_per_bag):
    feature_size = np.array(features).shape[1]
    interpolated_features = np.zeros((features_per_bag, feature_size))
    interpolation_indicies = np.round(np.linspace(0, len(features) - 1,
num=features_per_bag + 1))
    count = 0
    for index in range(0, len(interpolation_indicies)-1):
        start = int(interpolation_indicies[index])
        end = int(interpolation_indicies[index + 1])

        assert end >= start

        if start == end:
            temp_vect = features[start, :]
        else:
            temp_vect = np.mean(features[start:end+1, :], axis=0)

        temp_vect = temp_vect / np.linalg.norm(temp_vect)

        if np.linalg.norm(temp_vect) == 0:
            print("Error")

        interpolated_features[count,:]=temp_vect
        count = count + 1

    return np.array(interpolated_features)

def extrapolate(outputs, num_frames):
    extrapolated_outputs = []
    extrapolation_indicies = np.round(np.linspace(0, len(outputs) - 1,
```

```
num=num_frames))
    for index in extrapolation_indicies:
        extrapolated_outputs.append(outputs[int(index)])
    return np.array(extrapolated_outputs)

def preprocess_input(video):
    """Resize and subtract mean from video input
    Keyword arguments:
    video -- video frames to preprocess. Expected shape
        (frames, rows, columns, channels). If the input has more than 16 frames
        then only 16 evenly samples frames will be selected to process.
    Returns:
    A numpy array.
    """
    intervals = np.ceil(np.linspace(0, video.shape[0] - 1, 16)).astype(int)
    frames = video[intervals]

    reshape_frames = np.zeros((frames.shape[0], 171, 128, frames.shape[3]))
    for i, img in enumerate(frames):
        img = cv2.resize(img, (128, 171), interpolation = cv2.INTER_CUBIC)
        reshape_frames[i, :, :, :] = img

    mean = np.load('c3d_mean.npy')
    reshape_frames = reshape_frames[:, 8:120, 30:142, :]
    reshape_frames = np.expand_dims(reshape_frames, axis=0)

    return torch.from_numpy(reshape_frames).permute(0, 4, 1, 2, 3).double()

def get_video_clips(video_path):
    frames = get_video_frames(video_path)
```

```
    clips = sliding_window(frames, frame_count, frame_count)
    return clips, len(frames)

def get_video_frames(video_path):
    cap = cv2.VideoCapture(video_path)
    frames = []
    while (cap.isOpened()):
        ret, frame = cap.read()
        if ret == True:
            frames.append(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
        else:
            break
    return frames

for label in os.listdir('Test'):
    for video in os.listdir(f'Test/{label}'):
        video_name = video.split('.')[0]
        print(video_name)
        video_clips, num_frames = get_video_clips(f'Test/{label}/{video}')

        os.makedirs(f'Test_dataset/{label}', exist_ok=True)
        if f'{video_name}.npy' in os.listdir(f'Test_dataset/{label}'):
            continue

        rgb_features = []
        for i, clip in enumerate(video_clips):
            clip = np.array(clip)
            if len(clip) < 16:
                continue
```

```
            clip = preprocess_input(clip)
            rgb_feature = model(clip)
            rgb_features.extend(rgb_feature.detach().numpy())
        rgb_features = np.array(rgb_features)
        np.save(f'Test_dataset/{label}/{video_name}.npy', rgb_features)

class AnomalyDetector(nn.Module):
    def __init__(self):
        super(AnomalyDetector, self).__init__()
        self.fc1 = nn.Linear(487, 512)
        self.relu1 = nn.ReLU()
        self.dropout1 = nn.Dropout(0.6)

        self.fc2 = nn.Linear(512, 32)
        self.dropout2 = nn.Dropout(0.6)

        self.fc3 = nn.Linear(32, 1)
        self.sig = nn.Sigmoid()

        nn.init.xavier_normal_(self.fc1.weight)
        nn.init.xavier_normal_(self.fc2.weight)
        nn.init.xavier_normal_(self.fc3.weight)

    def forward(self, x):
        x = self.dropout1(self.relu1(self.fc1(x)))
        x = self.dropout2(self.fc2(x))
        x = self.sig(self.fc3(x))
        return x

class VideoAnalyzeApp(MDApp):
```

```
def init(self):
    super().init()
    self.sm = ScreenManager()
    self.sm.add_widget(Menu(name='menu'))
    self.sm.add_widget(Database(name='database'))
    self.sm.add_widget(Videodisplayer(name='videodisplayer'))

def open_menu(self):
    self.sm.current = 'menu'

def open_database_viewer(self):
    self.sm.current = 'database'

def open_video_displayer(self):
    self.sm.current = 'videodisplayer'

def open_database_video(self, text):
    self.sm.add_widget(DatabaseVideo(name='database_video', date=text))
    self.sm.current = 'database_video'

def build(self):
    return self.sm

Window.size = (1300, 750)
VideoAnalyzeApp().run()
cv2.destroyAllWindows()
```

ДОДАТОК З

Аналіз відеопотоку: ідентифікація аномальних подій

Опис програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ61186_20Б 13-1

Аркушів 8

Київ — 2020

АНОТАЦІЯ

Програмний продукт надає можливість переглядати проаналізований онлайн відеопоток, а також переглядати проаналізовані відео попередніх днів.

Система створена за допомогою мови програмування Python з використанням бібліотек PyTorch, Sqlalchemy, Kivy, OpenCV. В якості середовища розробки використовувалося PyCharm і Jupyter Notebook(для побудови моделі і аналізу).

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Розробка має назву «Система аналізу відеопотоку: ідентифікація аномальних подій».

Для коректної роботи системи аналізу відеопотоку необхідно встановити програмне забезпечення Docker, а також docker-compose і слідувати інструкціям, які описані в розділі 5.

Система написана за допомогою мови Python, база даних — PostgreSQL.

ФУНКЦІОНАЛЬНІ ПРИЗНАЧЕННЯ

Розроблена система створена для вирішення задачі ідентифікації аномальних подій з метою зменшення пограбувань, нападів, крадіжок, знущань, вбивств та інших кримінальних подій.

Дану систему реалізовано за допомогою:

— Згорткової нейронної мережі.

— Графічного інтерфейсу.

— Бази даних

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програмний продукт складається із трьох головних складових: нейронної мережі, графічного інтерфейсу, а також бази даних.

Принцип роботи наступний:

1) Як тільки запускається система, нейронна мережа починає свою роботу;
2) Береться певний часовий інтервал, ці дані проходять попередню оброку;
3) Дані подаються на вхід мережі;
4) Результат роботи мережі відобрається на екрані і записується до бази.

Взявши на оброку декілька секунд відео, ці дані конвертуються до картинок, проходять оброку даних для подачі на вхід мережі. Спочатку дані подаються на вхід згорткової мережі, виходом якої є дані для входу до класифікатора.

Отримавши результат класифікатора перевіряється чи отриманий клас співпадає із попереднім, якщо ні — відображаємо і записуємо до бази початок нового часового сегменту

Користувач може переглянути проаналізовані відео попередніх днів, нажавши кнопку «Database displayer». Конкретне відео, а також логи беруться із бази і відображаються на екран.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для роботи програмного продукту потрібний стаціонарний ПК або ноутбук із відеокартною яка оснащена відеопроцесором(можна і без, але рекомендується для стабільной, швидкої роботи).

Користувачам не потрібно встановлювати нічого, окрім інструменту Docker.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для системи є:

— Відеопоток.

Вихідними даними є:

— Клас, до якого належить сегмент відео.